Le PHP

Table des matières

1. Introduction

1.1. Descriptif

Avant de commencer à écrire vos pages en PHP, il faut d'abord connaître un minimum de choses sur ce langage.

PHP (officiellement, ce sigle est un acronyme récursif pour "PHP: Hypertext Preprocessor") est un langage de scripts généraliste, Open Source, et spécialement conçu pour le développement d'applications web. Il peut être intégré facilement à vos pages HTML.

Le code PHP que vous allez insérer dans vos pages WEB sera repéré par un serveur WEB (si il est muni de l'extension PHP) qu'il l'enverra à PHP pour l'interpréter (je parle bien d'interprétation et non de compilation).
PHP est supporté par de nombreux serveurs WEB, dont le fameux projet Apache.

Grâce à ces portions de code PHP que vous allez insérer dans vos pages WEB, PHP vous permettra d'écrire rapidement des pages WEB à contenus dynamiques. Surtout si il est couplé avec un serveur de bases de données relationnelles tel que MySQL.

1.2. Le fonctionnement

Le code PHP est inclus entre une balise de début et une balise de fin qui permettent au serveur web de passer en "mode PHP". La connaissance du code HTML est indispensable pour commencer ses premières pages en PHP.
Il faut savoir que lorsque vous insérez le moindre petit bout de code PHP dans une page HTML, vous devrez changer l'extension de ce fichier en .php.

Il est tout à fait possible de mélanger, au sein d'une même page WEB, des instructions HTML et des instructions PHP.
Les marques qui délimitent la portion de code PHP au sein du HTML, s'appellent des balises :

- **<?php** pour marquer le début d'une portion de code PHP
- **?>** pour marquer la fin d'une portion de code PHP

Les instructions du code PHP se placeront naturellement entre ces deux balises.

Exemple

```
<html>
<head>
<title>Test</title>
</head>

<body>
<p>un bout de code en HTML</p>
<?php
echo 'Mon premier script en PHP';
?>
</body>
</html>
```

Comme tout bon langage de programmation, PHP offre la possibilité de commenter son code. Pour cela, deux techniques :

- pour commenter une seule ligne de code PHP, //
- pour commenter une portion de code, **/*** au début et ***/** à la fin du commentaire

Exemple

```
<?php
// ceci est un commentaire sur une seule ligne

/* ceci est
un commentaire
sur plusieures lignes */
?>
```

Ce qui distingue le PHP des autres langages de script est que le code est exécuté sur le serveur. Si vous avez un script similaire sur votre serveur, le client ne reçoit que le résultat du script, sans aucun moyen d'avoir accès au code qui a produit ce résultat. Vous pouvez configurer votre serveur web afin qu'il analyse tous vos fichiers HTML comme des fichiers PHP. Ainsi, il n'y a aucun moyen de distinguer les pages qui sont produites dynamiquement des pages statiques.

1.3. Les besoins pour bien commencer

Pour bien commencer, il vous faut deux choses essentielles :

- un éditeur de texte pour écrire vos portions de code en PHP
- un environnement de développement afin de tester ces portions de code

2. EasyPHP

2.1. Introduction à EasyPHP

Afin de faire fonctionner PHP, il est nécessaire à la base d'en télécharger les sources depuis un site spécialisé (par exemple PHP.net), puis de compiler celui-ci (ainsi que d'éditer les liens) afin de créer un fichier exécutable.

Ce processus demande des notions avancées en informatique, c'est pourquoi trois adeptes de PHP (Emmanuel Faivre, Laurent Abbal et Thierry Murail) ont mis au point un package (appelé *EasyPHP*) contenant 3 produits incontournables de la scène PHP :

- Le serveur Web Apache
- Le moteur de scripts PHP4
- La base de données MySQL
- Un outil de gestion de base de donnée graphique, Phpmyadmin

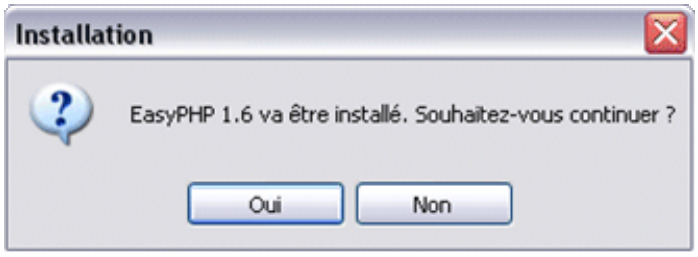
EasyPHP est ainsi un pack fonctionnant sous Windows permettant d'installer en un clin d'oeil les éléments nécessaires au fonctionnement d'un site web dynamique développé en PHP

2.2. Récupérer EasyPHP

Le pack EasyPHP est disponible sur le site suivant :

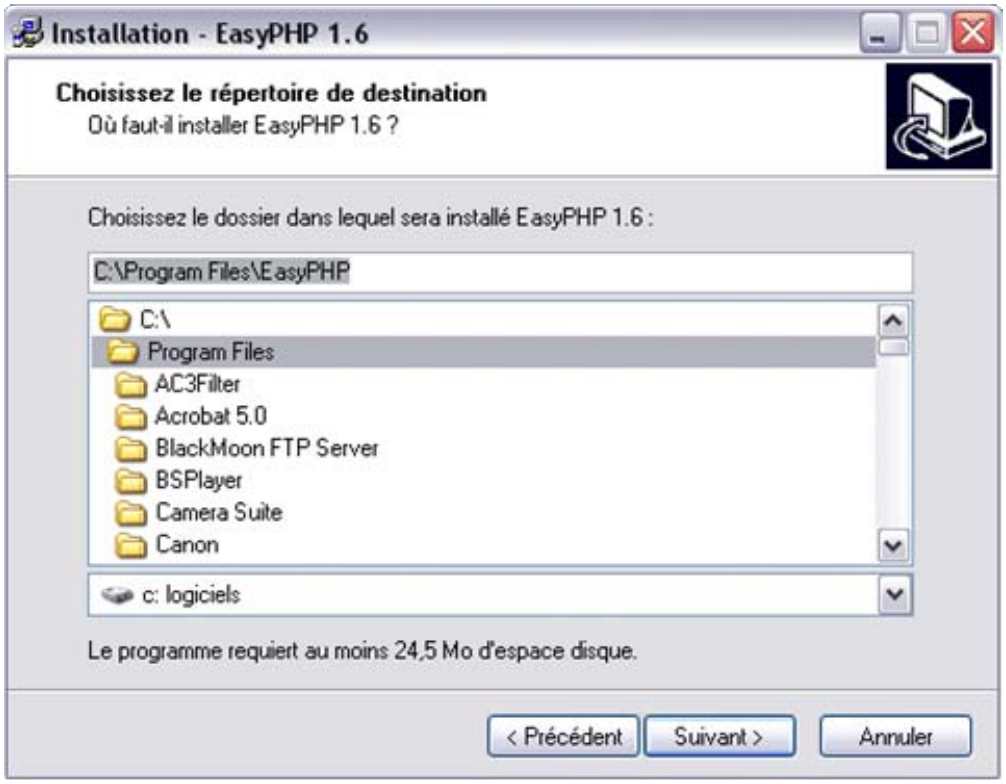
http://www.easyphp.org/telechargements.php3

Téléchargez le et installez le. Vous devriez alors voir la fenêtre suivante :



Accepter, et de même, prenez connaissance du contrat de licence.

Choisissez ensuite le répertoire d'installation d'EasyPHP (nous, nous avons laisser le répertoire proposé par l'installateur, c'est-à-dire le répertoire C:Program FilesEasyPHP).



Confirmez, l'installation devrait ensuite se poursuivre sans aucun problème.
Une fois l'installation achevée, vous devriez vous retrouver devant l'écran suivant :

Terminer l'installation en cliquant sur Terminer.

2.3. Démarrer EasyPHP

Pour démarrer Apache, MySQL et PHP, il vous suffit de lancer EasyPHP à partir du groupe créé dans le menu démarrer :

Pour vérifier si EasyPHP fonctionne, il vous suffit de taper dans votre navigateur préféré :

- http://localhost
- ou http://127.0.0.1

Les deux adresses ci-dessus représentant votre machine locale.

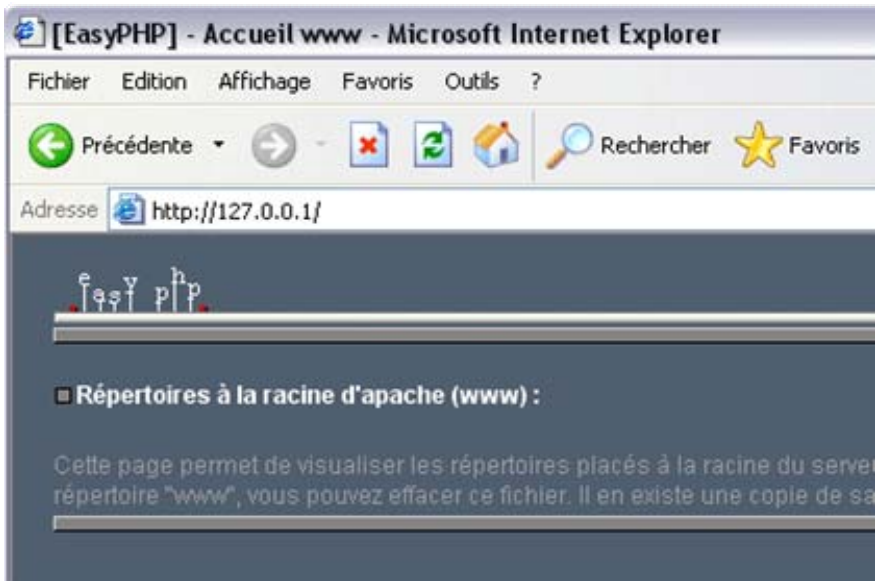
Une fois ceci fait, dans votre barre des taches Windows (en bas à droite), vous devriez avoir une icône représentant un E. En approchant votre souris dessus, une petite bulle d'aide devrait vous dire qu'EasyPHP est arrêté.

Si tel est le cas, faites un clic droit dessus, et dans le menu qui s'affiche, sélectionner l'option Démarrer.

Le petit E devrait maintenant vous informer que votre EasyPHP est en état de marche. Remarquer au passage, qu'à côté du E, un petit point rouge devrait clignoter.

Faites maintenant un clic droit (toujours sur le E) et dans le menu contextuel, choisissez l'option Web local.

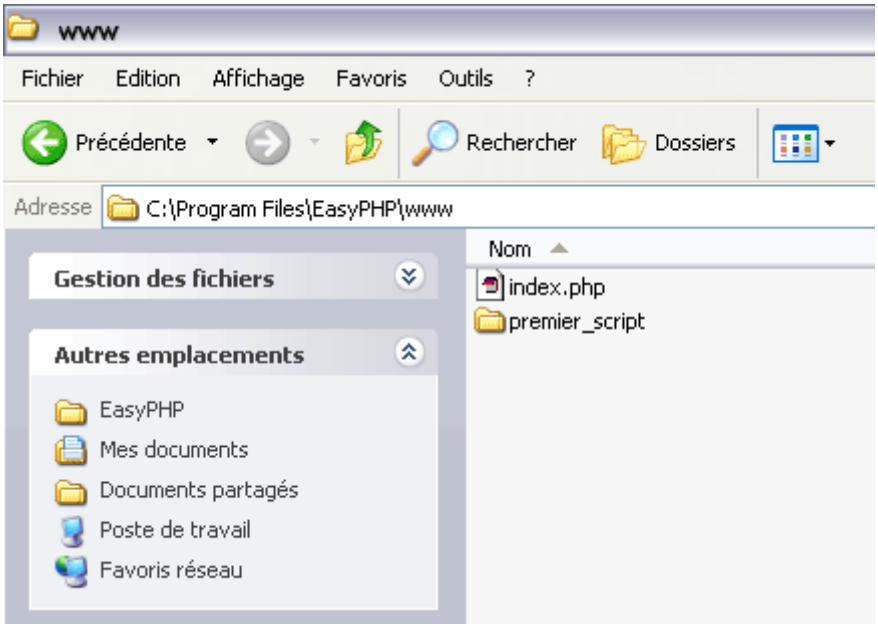
La fenêtre Web suivante devrait alors s'ouvrir sous vos yeux.
Noter qu'il n'y a aucun répertoire à la racine d'Apache, ce qui est normal puisque nous n'avons, pour l'instant, écrit aucun script PHP.



2.4. Editer votre site

Rendez-vous alors dans le répertoire C:\Program Files\EasyPHP\www.
Ne supprimer surtout pas le fichier index.

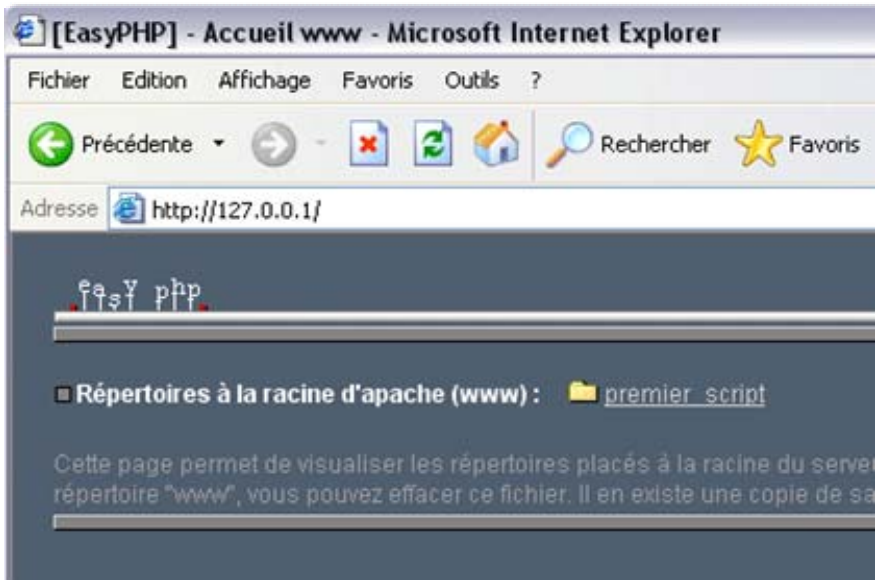
Créer alors un répertoire du nom de votre choix (nous, nous l'avons nommé premier_script).



Ouvrer alors ce répertoire, et dedans, créer un nouveau fichier PHP.
Ouvrer ce fichier, et placez y dedans, le code suivant :

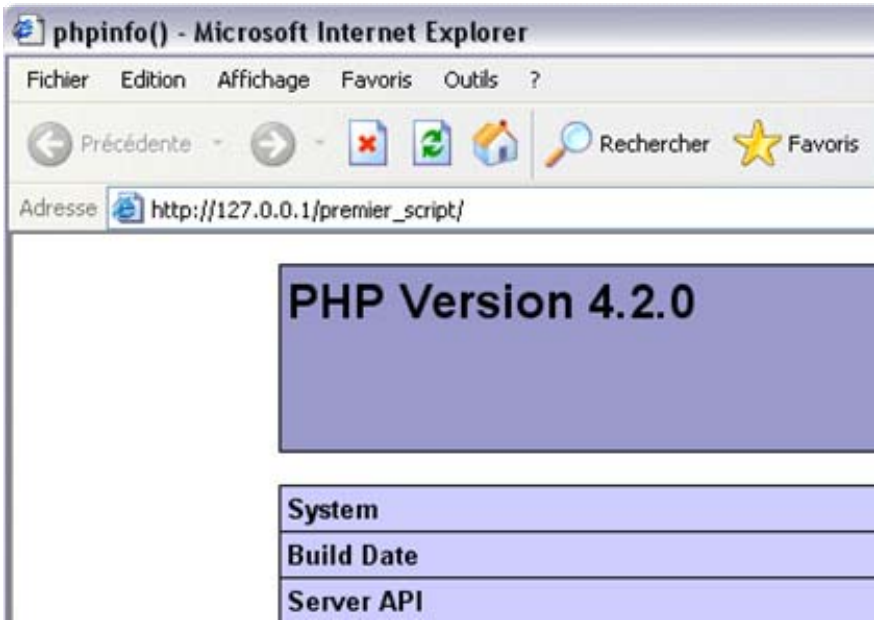
```
<?
phpinfo();
?>
```

Fermer votre fichier en le sauvegardant au passage, puis retourner à votre Web local.
Rafraîchissez votre page, et normalement, vous devriez avoir l'écran suivant :



En effet, c'est là que nous voyons l'intérêt du fichier index.php : en fait, le répertoire www est scanné à la recherche de tous les répertoires (vous devrez donc créer un répertoire pour chacun de vos projets PHP).

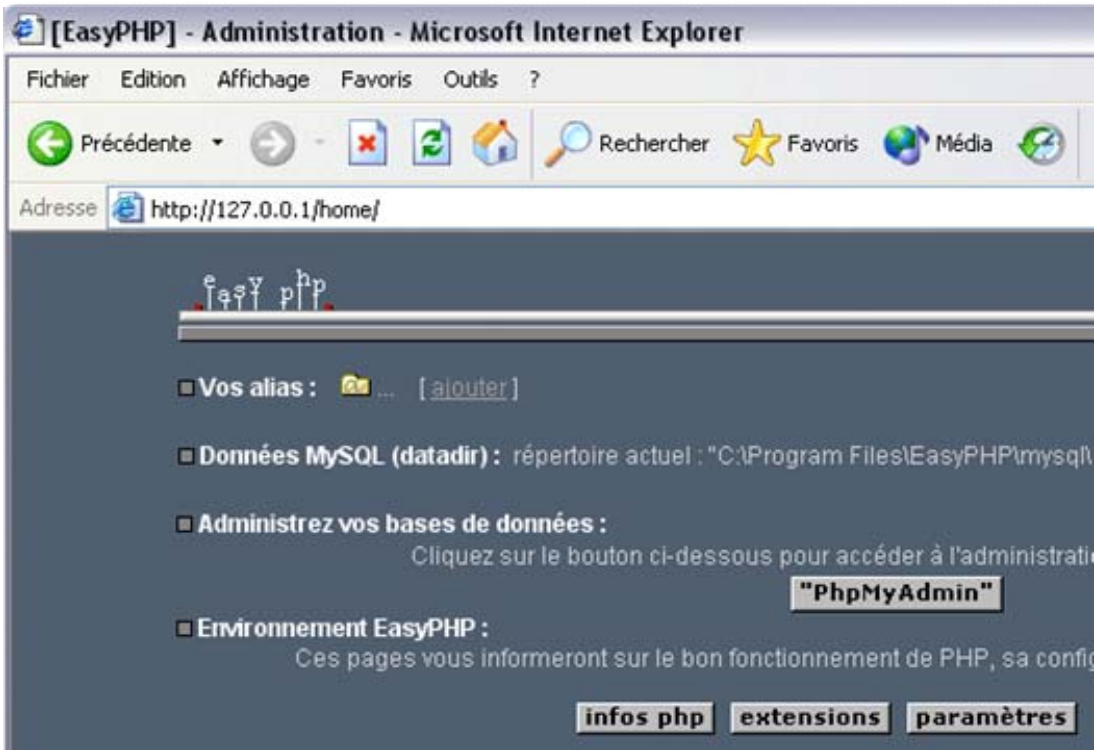
Cliquer alors sur le lien premier_script.
L'écran suivant devrait maintenant apparaître.



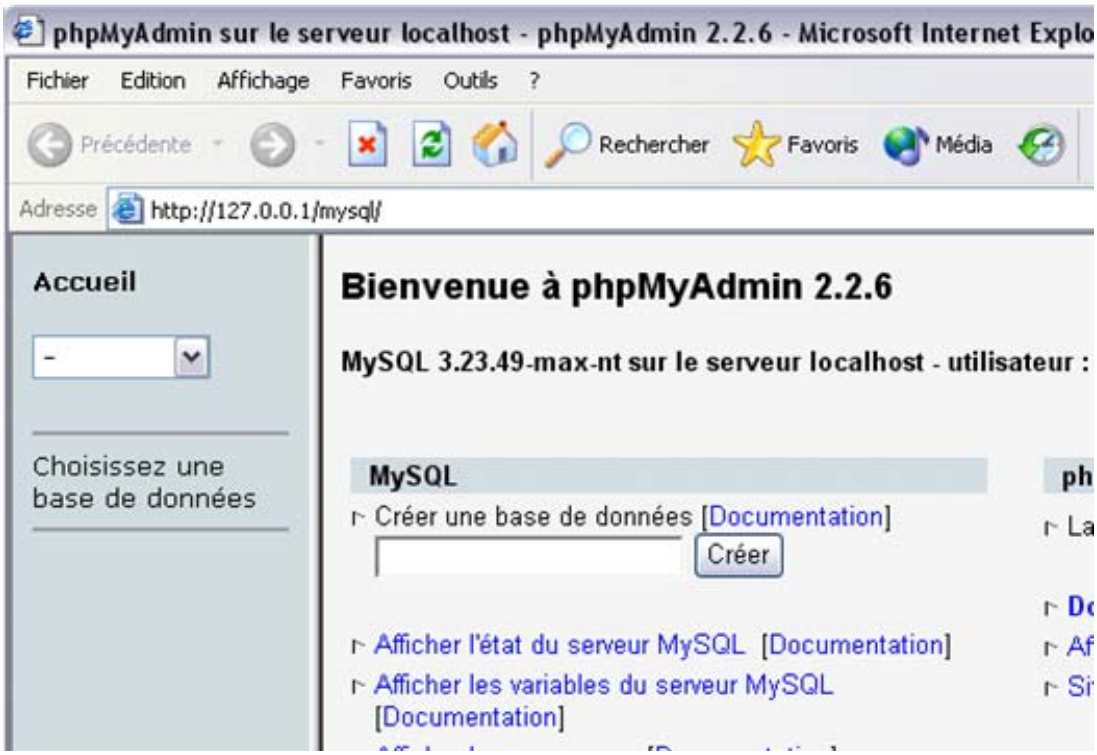
En ouvrant ce répertoire, Apache est parti à la recherche d'un fichier de type index.
Il l'a trouvé, et il s'est rendu compte que c'était un fichier php.
Il a alors passé la main à PHP qui s'est mis à interprété le code php contenu dans ce fichier (notre **phpinfo**()), ce qui a permis à Apache de nous afficher le résultat de l'interprétation du code PHP.

Faites maintenant un clic droit sur le E qui est dans la barre des taches.
Sélectionner l'option Administration dans le menu contextuel.
L'écran suivant devrait alors apparaître :

Cliquer sur PHPMyAdmin (afin de gérer toutes vos bases de données).



L'écran suivant apparaît alors.
Il s'agit de PHPMyAdmin, un script qui vous permettra d'administrer toutes vos bases de données sur une page WEB (ce qui est beaucoup plus pratique que de le faire directement en ligne de commande via MySQL).



Une fois votre utilisation terminée d'EasyPHP, n'oubliez surtout pas de l'éteindre !!!
Pour cela, faites un clic droit sur le E de votre barre des taches, et choisissez l'option Arrêter du menu contextuel.

Cela aura pour effet de couper votre serveur http (Apache) ainsi que votre serveur de bases de données (MySQL).

En effet, si vous ne coupez pas votre EasyPHP, il faut que vous sachiez que si votre port 80 est consultable de l'extérieur, n'importe qui peut à accéder à la racine de votre Apache, et donc de voir sur quoi vous travaillez en ce moment même.
Naturellement, cette personne doit connaître votre adresse IP pour pouvoir faire ceci.

Et voila, vous voici fin prêt à attaquer tous les autres cours de ce site, pour enfin, faire vos tous premiers scripts PHP :)

3. Déclarer des variables

Etudions dans un premier cours la déclaration des différents types de variables.

En PHP, les variables sont représentées par une chaîne de caractères, ayant toujours comme premier caractère, le caractère dollar ($). Les variables peuvent avoir n'importe quelle lettre en deuxième caractère du moment qu'il ne s'agit pas d'un chiffre. De plus, on ne peut mettre d'espace dans le nom d'une variable.

Puis, pour assigner une valeur à une variable, on tachera d'utiliser l'opérateur =, tout en prenant soin de toujours placer la variable qui reçoit le résultat d'une opération à gauche du signe =.

Remarque

- Lorsque l'on désire affecter une chaîne de caractères à une variable, il faut placer cette chaîne de caractères entre deux ".
- Lorsque l'on désire affecter une valeur numérique à une variable, il ne faut pas placer de " autour cette valeur (en fait, c'est possible de mettre des " autour d'une valeur numérique, mais ensuite, il faut être vraiment vigilent, car on pourrait faire la confusion entre une valeur numérique et une chaîne de caractères).

Exermples

```
<?phpphp
$nom = "LA GLOBULE";
// $nom contient alors la chaîne de caractères LA GLOBULE.
```

```
$mon_chiffre = 12;
// $mon_chiffre contient la valeur numérique 12.
```

```
$5toto = "test";
// Cette déclaration n'est pas valide car le nom de la variable commence
par un chiffre
?>
```

Voyons maintenant la déclaration des variables de type tableau (array).
Il s'agit d'une variable contenant différentes informations indicées.

Exemple 1

Supposons la variable $fruit de type array.
On pourrait alors avoir le code suivant :

```
<?php
$fruit[0] = "fraise";
$fruit[1] = "banane";
$fruit[2] = "abricot";
?>
```

Nous aurions eu le même résultat en exécutant le bout de code suivant :

En revanche, cette syntaxe est moins lisible, vu que souvent, on n'arrive plus vraiment à savoir à quelle page se trouve l'information recherchée (on s'emmêle dans les indices).

```
<?php
$fruit[] = "fraise";
$fruit[] = "banane";
$fruit[] = "abricot";
?>
```

Exemple 2

Au lieu d'utiliser des chiffres pour les indices (comme dans notre exemple où nous avons utilisé les indices 0, 1 et 2) nous pouvons très bien utiliser des chaînes de caractères.

```
<?php
$fruit['le_meilleur'] = "fraise";
$fruit['le_prefere_de_Julien'] = "banane";
$fruit['mon_prefere'] = "abricot";
?>
```

Ce qui pourrait alors donner :

Or dans ce cas, il faut évidement utiliser pour chaque indice du tableau, une chaîne de caractère unique.

4. Afficher le contenu des variables

Lors de cet exercice, nous allons mettre en pratique notre premier bout de code en PHP.
Il faut également savoir que toutes les variables en PHP commencent par le signe dollar ($), et il faut également ne pas oublier de placer un ; à chaque fin d'instruction PHP.

Etudions le code suivant :

```
<?php
$nom = "LA GLOBULE";
echo 'Bonjour ';
echo $nom;
echo ' !';
?>
```

Ce qui affichera à l'écran :

> Bonjour LA GLOBULE !

Au passage, remarquons la commande PHP **echo**(). Cette fonction nous permet d'afficher à l'écran des chaînes de caractères, qui peuvent être définies directement par l'utilisateur (echo 'Bonjour ';) ou qui peuvent être des contenus de variables (echo $nom;).

Attention

En effet, si nous avions écris le code suivant :

```
<?php
$nom = "LA GLOBULE";
echo 'Bonjour ';
echo '$nom';
echo ' !';
?>
```

Nous aurions eu à l'écran :

> Bonjour $nom !

Autre Exemple

Afficher la date et l'heure du jour.

```
<?php
$date_du_jour = date ("d-m-Y");
$heure_courante = date ("H:i");
echo 'Nous sommes le : ';
echo $date_du_jour;
echo ' Et il est : ';
echo $heure_courante;
?>
```

Ce qui affichera à l'écran :

Nous sommes le 17-09-2002 Et il est 12:10

Dans ce cas, nous venons d'utiliser la fonction **date**() qui nous permet d'afficher la date du jour ainsi que l'heure courante (en fait, la date du serveur). Dans un premier temps, nous avons affecté à la variable $date_du_jour le contenu que retourne la fonction **date**() munie des paramètres "d-m-Y", soit 17-09-2002, puis nous avons affecté à la variable $heure_courante le contenu que retourne la fonction **date**() munie des paramètres "H:i", soit 12:10.

Voici la liste des paramètres possibles pour la fonction **date**() :

a	"am" (matin) ou "pm" (après-midi)
A	"AM" (matin) ou "PM" (après-midi)
d	Jour du mois, sur deux chiffres (éventuellement avec un zéros) : "01" à "31"
D	Jour de la semaine, en trois lettres (et en anglais) : par exemple "Fri" (pour Vendredi)
F	Mois, textuel, version longue; en anglais, i.e. "January" (pour Janvier)
h	Heure, au format 12h, "01" à "12"
H	heure, au format 24h, "00" à "23"
g	Heure, au format 12h sans les zéros initiaux, "1" à "12"
G	Heure, au format 24h sans les zéros initiaux, "0" à "23"
i	Minutes; "00" à "59"
j	Jour du mois sans les zéros initiaux: "1" à "31"
l	Jour de la semaine, textuel, version longue; en anglais, i.e. "Friday" (pour Vendredi)
L	Booléen pour savoir si l'année est bissextile ("1") ou pas ("0")
m	Mois; i.e. "01" à "12"
n	Mois sans les zéros initiaux; i.e. "1" à "12"
M	Mois, en trois lettres (et en anglais) : par exemple "Jan" (pour Janvier)
s	Secondes; i.e. "00" à "59"
S	Suffixe ordinal d'un nom

5. Les variables prédéfinies

Voyons maintenant les variables d'environnements.

En effet, PHP propose toute une série de variables qui sont déjà présentes dans le langage sans que vous n'ayez à les déclarer. Ces variables s'écrivent toujours en majuscules et nous fournissent divers renseignements.

Voici la liste des variables d'environnement existantes :

Variable	Description
$_SERVER['DOCUMENT_ROOT']	Racine du serveur
$_SERVER['HTTP_ACCEPT_LANGUAGE']	Langage accepté par le navigateur
$_SERVER['HTTP_HOST']	Nom de domaine du serveur
$_SERVER['HTTP_USER_AGENT']	Type de navigateur
$_SERVER['PATH_INFO']	Chemin WEB du script
$_SERVER['PATH_TRANSLATED']	Chemin complet du script
$_SERVER['REQUEST_URI']	Chemin du script
$_SERVER['REMOTE_ADDR']	Adresse IP du client
$_SERVER['REMOTE_PORT']	Port de la requête HTTP
$_SERVER['QUERY_STRING']	Liste des paramètres passés au script
$_SERVER['SERVER_ADDR']	Adresse IP du serveur
$_SERVER['SERVER_ADMIN']	Adresse de l'administrateur du serveur
$_SERVER['SERVER_NAME']	Nom local du serveur
$_SERVER['SERVER_SIGNATURE']	Type de serveur
$_SERVER['REQUEST_METHOD']	Méthode d'appel du script

Ces variables peuvent être utilisées n'importe quand dans vos scripts.
Voici un exemple où vous pouvez afficher l'adresse IP de la personne qui se connecte sur

```
<?php
echo 'Votre adresse IP est : '.$_SERVER['REMOTE_ADDR'];
?>
```

votre site :

Ce qui affichera à l'écran :

Votre adresse IP est : 80.12.45.26

6. Concaténer deux chaînes

Qu'est la concaténation de chaîne de caractères ?

Exemple

Prenons un exemple simple avec deux chaînes de caractères :

- la première chaîne de caractères sera : "J'apprend "
- la seconde chaîne sera : "le PHP"

En faisant une concaténation de ces deux chaînes, nous obtiendrons la chaîne suivante :

> "J'apprend le PHP".

Tachons maintenant de mettre alors en évidence l'importance de la concaténation de chaîne de caractères avec l'exemple de l'exercice précédent.

On avait alors comme code PHP :

```
<?php
$nom = "LA GLOBULE";
echo 'Bonjour ';
echo $nom;
echo ' !';
?>
```

Or, les trois lignes avec l'instruction **echo**() peuvent se simplifier en une seule grâce à la concaténation. En PHP, la concaténation de chaîne s'effectue grâce au point.

On a alors :

```
<?php
$nom = "LA GLOBULE";
echo 'Bonjour '.$nom.' !';
?>
```

Ce qui affichera à l'écran :

> Bonjour LA GLOBULE !

7. Les structures de contrôles

Apres avoir vu un premier aperçu du langage PHP, nous allons maintenant étudier les différentes structures de contrôles du langage. Les structures de contrôles nous permettrons de faire des tests entre les variables et d'exécuter diverses boucles.

Voici un petit récapitulatif des principales structures de contrôles :

Instruction	Signification
if	Si
else	Sinon
elseif	Sinon si
switch	Selon
for	Pour chaque (boucle)
while	Tant que (boucle)
==	Strictement égal
!=	Différent
<	Strictement inférieur
>	Strictement supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal
and ou &&	ET logique
or ou ||	OU logique

7.1. If Elseif Else

Exemple

```
<?php
$nombre = 11 ;
if ($nombre >= 0 && $nombre < 10) {
    // on teste si la valeur de notre variable est comprise entre 0 et 9
    echo $nombre.' est compris entre 0 et 9';
}
elseif ($nombre >= 10 && $nombre < 20) {
    // on teste si la valeur de notre variable est comprise
      entre 10 et 19
    echo $nombre.' est compris entre 10 et 19';
}
else {
    // si les deux tests précédents n'ont pas aboutis, alors on tombe
      dans ce cas
    echo $nombre.' est plus grand que 19';
}
?>
```

A l'affichage on aura :

> 11 est compris entre 10 et 19

7.2. Switch

Le switch représente une succession d'un if et de plusieurs elseif. En revanche, utiliser un switch à un certain avantage comparé au if, c'est que sa structure est beaucoup moins lourde et nettement plus agréable à lire.

Prenons un exemple simple. Nous allons déclarer une variable contenant une chaîne de caractères, puis nous allons tester cette chaîne grâce au switch.

```
<?php
$nom = "LA GLOBULE";

switch ($nom) {
    case 'Jean' :
        echo 'Votre nom est Jean.';
        break;
    case 'Benoît' :
        echo 'Votre nom est Benoît.';
        break;
    case 'LA GLOBULE' :
        echo 'Votre nom est LA GLOBULE.';
        break;
    default :
        echo 'Je ne sais pas qui vous êtes !!!';
}
?>
```

Dans notre cas, vu que $nom contient la chaîne de caractère LA GLOBULE, on verra alors s'afficher à l'écran la phrase suivante :

> Votre nom est LA GLOBULE.

En revanche, si la variable $nom avait contenu la chaîne de caractère "toto", ce même code aurait affiché à l'écran :

> Je ne sais pas qui vous êtes !!!

En utilisant un if puis une succession de elseif, le code suivant aurait exactement eu le même

```php
<?php
$nom = "LA GLOBULE";

if ($nom == "Jean") {
    echo 'Votre nom est Jean.';
}
elseif ($nom == "Benoît") {
    echo 'Votre nom est Benoît.';
}
elseif ($nom == "LA GLOBULE") {
    echo 'Votre nom est LA GLOBULE.';
}
else {
    echo 'Je ne sais pas qui vous êtes !!!';
}
?>
```

Attention

Notez bien l'utilisation de break dans chaque cas de votre switch. Si celui-ci est omis, tous les messages s'afficheront.

7.3. For

La structure de contrôle for nous permet d'écrire des boucles.

Prenons l'exemple suivant :

```php
<?php
$chiffre = 5;

// Début de la boucle
for ($i=0; $i < $chiffre; $i++) {
    echo 'Notre chiffre est différent de '.$i.'<br />';
}
// Fin de la boucle

echo 'Notre chiffre est égal à '.$i;
?>
```

Ce qui affichera à l'écran :

Notre chiffre est différent de 0
Notre chiffre est différent de 1
Notre chiffre est différent de 2
Notre chiffre est différent de 3
Notre chiffre est différent de 4
Notre chiffre est égal à 5

7.4. while

Voyons maintenant la boucle while.
Cette boucle peut toujours être remplacée par une boucle for.

Reprenons l'exemple précédent, et écrivons le à l'aide de la boucle while, on a :

```
<?php
$chiffre = 5;
$i = 0;

// Début de la boucle
while ($i < $chiffre) {
    echo 'Notre chiffre est différent de '.$i.'<br />';
    $i = $i + 1;
}
// Fin de la boucle

echo 'Notre chiffre est égal à '.$i;
?>
```

Ce programme affichera la même chose le code que l'on a utilisé pour la boucle for.

8. Lire et écrire dans un fichier texte

Tentons maintenant de lire et d'écrire dans un fichier texte, fichier se trouvant sur votre serveur FTP. Afin de mettre en pratique cet exercice, créer un fichier donnees.txt que vous placerez dans le même répertoire que le script PHP.

Supposons que ce fichier texte contienne la ligne suivante :

> "Salut à tous :)".

Soit alors, le code PHP suivant :

```
<?php

// Ouverture du fichier
$fp = fopen ("donnees.txt", "r");

// Lecture des 255 premiers caractères du fichier
$contenu_du_fichier = fgets ($fp, 255);

// Fermeture du fichier
fclose ($fp);

// Affichage
echo 'Notre fichier contient : '.$contenu_du_fichier;
?>
```

Ce qui affichera à l'écran :

> Notre fichier contient : Salut à tous :)

Aparté

Vous n'êtes absolument pas obligé de mettre ce fichier texte dans le même répertoire que le script PHP. En effet, si votre script PHP est à la racine de votre site et que le fichier à lire se trouve dans le répertoire /toto, vous écrirez alors votre script de la manière suivante :

```
<?php
$fp = fopen ("toto/donnees.txt", "r");
$contenu_du_fichier = fgets ($fp, 255);
fclose ($fp);
echo 'Notre fichier contient : '.$contenu_du_fichier;
?>
```

Ce qui produira un résultat tout a fait identique.

Etudions maintenant tous les paramètres possibles de la fonction **fopen**() :

r	le fichier est ouvert en lecture et le pointeur est positionné au début du fichier.
r+	le fichier est ouvert en lecture / écriture et le pointeur est positionné au début du fichier.
w	le fichier est ouvert en écriture et le pointeur est positionné au début du fichier. le fichier est écrasé s'il existe sinon il est créé.
w+	le fichier est ouvert en lecture / écriture. Le fichier est écrasé s'il existe sinon il est créé.
a	le fichier est ouvert en écriture et le pointeur est positionné à la fin du fichier. si le fichier n'existe pas il est créé.
a+	le fichier est ouvert en lecture / écriture et le pointeur est positionné à la fin du fichier. si le fichier n'existe pas il est créé.
b	permet d'ouvrir un fichier en mode binaire.

Un exemple concret : un mini compteur du nombre de visites.

Tout d'abord créez un fichier compteur.txt dans le même répertoire que le script qui va suivre. Placez le chiffre "0" dans ce fichier.

Soit alors le bout de code PHP suivant :

```
<?php

// Ouverture du fichier
$fp = fopen ("compteur.txt", "r+");

// Lecture du fichier
$nb_visites = fgets ($fp, 11);

// Incrémentation du nombre de visites
$nb_visites = $nb_visites + 1;

// Placer le pointeur a début du fichier
fseek ($fp, 0);

// Ecrire la nouvelle valeur
fputs ($fp, $nb_visites);

// Fermeture du fichier
fclose ($fp);

// Affichage
echo 'Ce site compte '.$nb_visites.' visiteurs !';
?>
```

9. Récupérer les données des formulaires

Voyons maintenant comment créer des formulaires, et les utiliser. Les formulaires vont permettre à vos visiteurs de soumettre des informations, que ce soit un nom, un prénom, un chiffre, etc...

Prenons le code suivant :

```
<html>
<head>
<title>Ma page de test</title>
</head>
<body>
<form action = "traitement.php" method="post">
Votre nom : <input type = "text" name = "nom"><br />
Votre fonction : <input type = "text" name = "fonction"><br />
<input type = "submit" value = "Envoyer">
</form>
</body>
</html>
```

Ce qui donnera à l'écran :

Votre nom :

Votre fonction :

Envoyer

Lorsque l'utilisateur cliquera sur le bouton "Envoyer", les données du formulaire seront envoyées sur la page traitement.php.

Et dans la page traitement.php, nous allons récupérer une variable de type tableau ($_POST : car notre formulaire a comme method la valeur post).

En clair, dans la page traitement.php, on aura une variable $_POST['nom'] qui contiendra la chaîne de caractères qu'aura saisi le visiteur dans le champ "Votre nom : " (on a la variable $_POST['nom'], car dans l'attribut name de notre formulaire pour le champ concernant le nom).

De même, on aura une variable $_POST['fonction'] qui contiendra la chaîne de caractères qu'aura saisi le visiteur dans la champ "Votre fonction : " (encore une fois, on a la variable $_POST['fonction'] car l'attribut name du champ prend la valeur fonction).

Prenons ensuite le code suivant pour la page traitement.php :

```
<html>
<head>
<title>Ma page de traitement</title>
</head>
<body>
<?
// on teste la déclaration de nos variables
if (isset($_POST['nom']) && isset($_POST['fonction'])) {
    // on affiche nos résultats
    echo 'Votre nom est '.$_POST['nom'].' et votre fonction est
'.$_POST['fonction'];
}
?>
</body>
</html>
```

En supposant que l'on écrive "LA GLOBULE" dans le champ "Votre nom" et "Webmaster" dans le champ "Votre fonction", on verra alors s'afficher à l'écran :

Votre nom est LA GLOBULE et votre fonction est Webmaster

Dans le cas où le formulaire utilise une méthode get, nous utilisons la variable tableau $_GET.

Voyons maintenant le cas des formulaires munis d'un champ de type file (formulaire permettant le téléchargement de fichiers sur votre site).
Imaginons que l'on ai le formulaire suivant :

```
<html>
<head>
<title>Ma page de test</title>
</head>
<body>
<form action = "traitement.php" method="post">
Votre fichier : <input type = "file" name = "mon_fichier"><br />
<input type = "hidden" name="MAX_FILE_SIZE" value="20000">
<input type = "submit" value = "Envoyer">
</form>
</body>
</html>
```

Pour récupérer votre fichier, vous avez à votre disposition le tableau $_FILES qui aura plusieurs entrées :

$_FILES['mon_fichier']['tmp_name']	le nom temporaire du fichier sur le serveur
$_FILES['mon_fichier']['name']	le nom original du fichier sur la machine cliente

$_FILES['mon_fichier']['type']	le type MIME du fichier
$_FILES['mon_fichier']['size']	la taille du fichier

Naturellement, vous pourrez utiliser ces valeurs pour tester votre fichier.
Si il correspond à vos attentes, vous pourrez finaliser votre téléchargement à l'aides des fonctions copy ou move_uploaded_file (afin de copier le fichier téléchargé sur le disque dur du serveur).

10. Les fonctions utilisateurs

PHP comprend, de base, une liste assez impressionnante de fonctions mises à votre disposition. En revanche, vous aussi, vous pouvez très bien écrire vos propres fonctions.

Nous allons donc écrire une fonction qui va nous permettre d'écrire un texte en gras, tout en spécifiant la couleur de ce texte, ainsi que sa taille.

On a alors le code suivant :

```
<?php
function affichage_texte ($taille, $couleur, $texte) {
    echo '<font size = "'.$taille.'" color =
"'.$couleur.'">'.$texte.'</font>';
}
?>
```

Placons ce code dans un fichier vierge nommé fonctions.php.

Soit ensuite le code du fichier index.php :

```
<?php
// on inclut le code de fonctions.php, donc le code de notre fonction
include ('fonctions.php');

// on affiche un texte
affichage_texte ("2", "red", "Mon texte");
?>
```

Ce qui affichera à l'écran :

Mon texte

11. Introduction aux bases de données

Prenons un exemple simple et concret : supposons que l'on désire développer une base de données contenant une liste de CD audio. Cette liste de CD sera en fait composée de tous les CD que possède chaque personne d'un groupe d'amis. Et ceci, afin de pouvoir se prêter mutuellement les différents CD, et de savoir exactement qui à quoi comme CD.

On suppose que le groupe d'amis est composé de 3 personnes :
- LA GLOBULE
- Jeremy
- Benoît

Chaque personne a un numéro de téléphone, et chaque personne possède un certain nombre de CD. On prendra aussi en considération le titre de l'album et le nom de l'interprète.

On a la base de données suivante :

Propriétaire	**N. tél**	**Auteur**	**Titre**
LA GLOBULE	06-48-85-20-54	Cassius	Au rêve
LA GLOBULE	06-48-85-20-54	Daft Punk	Discovery
Jeremy	06-48-74-26-01	Cassius	Au rêve
Jeremy	06-48-74-26-01	Télépopmusik	Genetic world
Benoît	06-47-01-59-36	Clamaran	Release yourself

Notez bien que ce tableau, en terme de base de données, se nomme une table et que chaque ligne du tableau se nomme un tuple. La première ligne du tableau comporte les attributs de la table (Propriétaire, N. tél, Auteur et Titre sont les attributs de notre table).

Faisons maintenant quelques interrogations sur cette base de données :

Qui possède un album de Cassius ?
>> **réponse** : LA GLOBULE et Jeremy

Quel est le numéro de téléphone de Benoît ?
>> **réponse** : 06-47-01-59-36

Quels sont les albums des Daft Punk disponibles dans la liste de CD ?
>> **réponse** : Discovery (il n'y en a qu'un seul)

Imaginons maintenant que Jeremy vienne de se faire voler son tout nouveau portable dans le métro, et qu'il change alors naturellement de numéro. Son nouveau numéro est alors 06-85-

98-78-12. Il vient de s'acheter un nouveau CD : Paradise de Bob Sinclar.
On insère alors une nouvelle ligne dans notre table, et l'on obtient donc :

Propriétaire	**N. tél**	**Auteur**	**Titre**
LA GLOBULE	06-48-85-20-54	Cassius	Au rêve
LA GLOBULE	06-48-85-20-54	Daft Punk	Discovery
Jeremy	06-48-74-26-01	Cassius	Au rêve
Jeremy	06-48-74-26-01	Télépopmusik	Genetic world
Benoît	06-47-01-59-36	Clamaran	Release yourself
Jeremy	06-85-98-78-12	Bob Sinclar	Paradise

Imaginons maintenant que j'interroge à nouveau ma base de données.

Quel est le numéro de téléphone de Jeremy ?
>> **réponse** : 06-85-98-78-12 ou bien 06-48-74-26-01

Problème majeur :
Jeremy possède deux numéros de téléphone.

Pour remédier a ce problème, il faudrait mettre à jour le numéro de téléphone de Jeremy dans toute la base. Dans notre cas, cette solution n'est pas vraiment gênante, en revanche, lorsque la table comporte quelques centaines voir milliers de t-uples, c'est déjà beaucoup plus gênant.

Ce problème survient généralement à cause d'une mauvaise conception de la base de données.

En effet, au lieu de créer une seule table contenant toutes les informations, nous aurions du créer deux tables :

- une contenant la liste des CD (Auteur et Titre)
- une contenant les informations des propriétaires des CD (Propriétaire et N. de tel)

Ensuite, il nous resterait a faire un lien entre les tables, nous permettant de savoir qui possède tel ou tel CD.

Mettons cette solution en pratique. On a alors notre table contenant la liste des propriétaires qui aura les attributs suivants :
- numéro du propriétaire
- nom du propriétaire
- numéro de téléphone du propriétaire

Et la table contenant la liste de CD, aura les attributs suivants :
- numéro du propriétaire du CD
- Auteur du CD
- Titre du CD

On aura alors :

N. du propriétaire	Propriétaire	N. tél
1	LA GLOBULE	06-48-85-20-54
2	Jeremy	06-85-98-78-12
3	Benoît	06-47-01-59-36

N. du propriétaire	Auteur	Titre
1	Cassius	Au rêve
1	Daft Punk	Discovery
2	Cassius	Au rêve
2	Télépopmusik	Genetic world
3	Clamaran	Release yourself
2	Bob Sinclar	Paradise

On remarque facilement que si une personne change de numéro de téléphone, et bien nous avons qu'une seule modification à effectuer (vu que chaque numéro de téléphone n'apparaît qu'une fois dans toute la base).

Dans ce cas, c'est le N. de propriétaire qui effectue la liaison entre les deux tables (c'est à dire la jointure).

Conclusion

Faites extrêmement attention au moment où vous créer les tables de votre base de données afin de ne pas se retrouver dans une situation où tout retour en arrière serait impossible : visualiser bien votre idée, écrivez sur papier ce dont vous avez réellement besoin pour votre base de données, et tentez au maximum d'éviter d'avoir des redondances dans vos tables.

12. Fonctions PHP pour MySQL

Nous venons de voir comment créer nos tables SQL à l'aide d'un PHPMyAdmin.

Avant de voir comment faire pour insérer, modifier, supprimer et afficher des tuples de notre base de données, il est bon de connaître les fonctions PHP permettant de manoeuvrer ces tuples.

Fonction	Signification
mysql_close	Ferme la connexion à une base de données
mysql_connect	Etablit une connexion vers la base de données spécifiée dans les arguments
mysql_error	Retourne la description textuelle d'une erreur générée par une action sur une base de données
mysql_fetch_array	Retourne un tableau qui représente tous les tuples sélectionnés (un indice du tableau correspond à un attribut des tuples obtenus). Chaque appel récupère le tuple suivant jusqu'à ce qu'il n'y en ait plus
mysql_free_result	Libère la mémoire associé à la requête spécifiée
mysql_num_rows	Retourne le nombre de tuple dans un résultat
mysql_query	Permet d'exécuter une requête SQL sur une base de données
mysql_select_db	Sélectionne la base de données par défaut

Voyons maintenant comment faire pour se connecter à une base de données. En effet, afin de pouvoir utiliser tous les éléments contenus dans une base de données, vous devez indiquer, sur toutes vos pages PHP où vous utilisez votre base, différents paramètres de connexion à votre base.

Etudions le code suivant :

```
<?php
$base = mysql_connect ('mon_serveur', 'login', 'password');
mysql_select_db ('ma_base_de_donnees', $base) ;
?>
```

La chaîne de caractères **mon_serveur** doit être remplacé par celle qui correspond au nom de votre serveur (en règle générale, il s'agit de localhost ; si ce n'est pas le cas, veuillez contacter votre hébergeur pour de plus amples informations). **login** correspond à votre login pour accéder à votre base. **password**, votre mot de passe. Et **ma_base_de_donnees** correspond au nom de votre base de données.

Grâce à ce code nous allons donc pouvoir effectuer toutes nos requêtes SQL sur les tables de notre base de données fraîchement créée.

Attention

Ce code doit toujours être présent avant toute opération sur votre base de données
(une seule fois par page suffit par contre).

Conseil

Faites vous un fichier connect_base.php.inc ou apparaîtra seulement ce morceau de code, et dans chaque page ou vous souhaitez utiliser votre base de données, vous n'aurez alors qu'à **include**() ce fichier de connexion.

13. Afficher les données de votre base

Maintenant que les tables de votre base de données sont créées, nous allons pouvoir interroger cette base de données, et par conséquent afficher les résultats sur vos pages WEB :)

Avant de plonger à l'inconnu dans le code PHP pour faire ces interrogations, nous allons voir comment s'effectuent ces interrogations par le biais de requêtes SQL.

Je vous rappelle que l'on avait alors deux tables qui peuvent être représentées ainsi :

La table liste_proprietaire :

N. du propriétaire	**Propriétaire**	**N. tél**
1	LA GLOBULE	06-48-85-20-54
2	Jeremy	06-85-98-78-12
3	Benoît	06-47-01-59-36

La table liste_disque :

N. du propriétaire	**Auteur**	**Titre**
1	Cassius	Au rêve
1	Daft Punk	Discovery
2	Cassius	Au rêve
2	Télépopmusik	Genetic world
3	Clamaran	Release yourself
2	Bob Sinclar	Paradise

Nous allons alors interroger la table pour connaître par exemple le numéro de téléphone de LA GLOBULE.

SELECT telephone
FROM liste_proprietaire
WHERE nom="LA GLOBULE";

Etudions maintenant le cas ou l'on effectue une sélection lorsque l'on doit effectuer une jointure entre deux tables. Interrogeons alors notre base de données pour connaître le nom des propriétaires de l'album Au rêve de Cassius.

SELECT liste_proprietaire.proprietaire
FROM liste_proprietaire, liste_disque

WHERE liste_disque.auteur = "Cassius"
AND liste_disque.titre = "Au rêve"
AND liste_proprietaire.numero = liste_disque.numero
ORDER BY liste_proprietaire.proprietaire **ASC**;

Remarque

Nous avons que dans nos requêtes SQL nous pouvions imposer au SGBD de ne sélectionner que les t-uples dont on impose la valeur de certains attributs (comme par exemple en imposant que l'attribut auteur soit égal à Cassius par le biais de la ligne **WHERE** liste_disque.auteur = "Cassius").

En revanche, nous pouvons également faire une recherche en n'imposant pas réellement la valeur de l'attribut mais plutôt en ne sélectionnant que les t-uples dont l'attribut commence par une certaine chaîne de caractères ou bien même de ne sélectionner que les t-uples dont l'attribut ne fait que contenir une chaîne de caractères. Tout ceci se fera grâce à la clause LIKE.

Exemple

SELECT liste_proprietaire.proprietaire
FROM liste_proprietaire, liste_disque
WHERE liste_disque.auteur **LIKE** "C%"
AND liste_disque.titre = "Au rêve"
AND liste_proprietaire.numero = liste_disque.numero
ORDER BY liste_proprietaire.proprietaire **ASC**;

Dans ce cas, nous n'avons plus la ligne **WHERE** liste_disque.auteur = "Cassius" mais la ligne **WHERE** liste_disque.auteur **LIKE** "c%".
Ce changement implique que nous allons choisir non pas les disques dont l'auteur est Cassius mais les disques dont l'auteur commence par la lettre c.

On aurait également pu faire :

SELECT liste_proprietaire.proprietaire
FROM liste_proprietaire, liste_disque
WHERE liste_disque.auteur **LIKE** "%s%"
AND liste_disque.titre = "Au rêve"
AND liste_proprietaire.numero = liste_disque.numero
ORDER BY liste_proprietaire.proprietaire **ASC**;

Et dans ce cas, nous aurions sélectionné les t-uples dont l'attribut auteur de la table liste_disque contient la lettre s.

Créons une page PHP nous permettant de réaliser exactement la même requête que la première de ce tutorial, c'est-à-dire la sélection du numéro de téléphone de LA GLOBULE.

On a alors le code suivant :

```
<?php
// on se connecte à notre base
$base = mysql_connect ('serveur', 'login', 'pass');
mysql_select_db ('ma_base', $base) ;
?>
<html>
<head>
<title>Numéro de téléphone de LA GLOBULE</title>
</head>
<body>
<?
// lancement de la requete
$sql = 'SELECT telephone FROM liste_proprietaire WHERE nom = "LA
GLOBULE"';

// on lance la requête (mysql_query) et on impose un message d'erreur si
la requête ne se passe pas bien (or die)
$req = mysql_query($sql) or die('Erreur SQL !<br />'.$sql.'<br
/>'.mysql_error());

// on recupere le resultat sous forme d'un tableau
$data = mysql_fetch_array($req);

// on libère l'espace mémoire alloué pour cette interrogation de la base
mysql_free_result ($req);
mysql_close ();
?>
Le numéro de téléphone de LA GLOBULE est :<br />
<? echo $data['telephone']; ?>
</body>
</html>
```

Ce qui affichera à l'écran :

> Le numéro de téléphone de LA GLOBULE est :
> 06-48-85-20-54

Mettons maintenant dans le cas où l'interrogation de la base de données ne retourne pas un, mais un certain nombre de tuples (nombre que l'on ne connaît pas).
En effet, recherchons tous les noms de propriétaires de disque, ainsi que leur numéro de téléphone.

On aura alors le code suivant :

```php
<?php
// on se connecte à notre base
$base = mysql_connect ('serveur', 'login', 'pass');
mysql_select_db ('ma_base', $base) ;
?>
<html>
<head>
<title>Nom et tél des membres</title>
</head>
<body>
<?
// lancement de la requête (on impose aucune condition puisque l'on
désire obtenir la liste complète des propriétaires
$sql = 'SELECT telephone, nom FROM liste_proprietaire';

// on lance la requête (mysql_query) et on impose un message d'erreur si
la requête ne se passe pas bien (or die)
$req = mysql_query($sql) or die('Erreur SQL !<br />'.$sql.'<br
/>'.mysql_error());

// on va scanner tous les tuples un par un
while ($data = mysql_fetch_array($req)) {
        // on affiche les résultats
        echo 'Nom : '.$data['nom'].'<br />';
        echo 'Son tél : '.$data['telephone'].'<br /><br />';
}
mysql_free_result ($req);
mysql_close ();
?>
</body>
</html>
```

Et ainsi, grâce à la boucle while, nous pouvons parcourir tous les tuples obtenus par la requête SQL.

Pour finir, nous pouvons juste dire que lorsque l'on effectue une sélection qui contient une jointure, le principe reste exactement le même.

Que faire en plus ?

Afin d'améliorer vos sélections, vous pouvez faire dépendre vos sélections du résultat obtenu par un formulaire.
En effet, imaginons une première page avec un formulaire nous permettant de choisir le nom d'un propriétaire via un menu déroulant. Ensuite, dans la page où vous allez faire votre requête (qui donc être également la page contenue dans le champ action de votre formulaire), vous allez récupérer une variable, par exemple $_POST['nom_proprio'] (cf. le tutorial sur la récupération des données par le biais des formulaires).
De plus, imaginons que l'on désire retrouver le numéro de téléphone de ce propriétaire (celui choisi dans le menu-déroulant).

On aura alors (page pointée par le champ action du formulaire):

```php
<?php
// on se connecte à notre base
$base = mysql_connect ('serveur', 'login', 'pass');
mysql_select_db ('ma_base', $base) ;
?>
<html>
<head>
<title>Numéro de téléphone du membre choisi</title>
</head>
<body>
<?
// on teste si notre variable est déclarée
if (isset($_POST['nom_proprio'])) {

        // lancement de la requête
        $sql = 'SELECT telephone FROM liste_proprietaire WHERE nom =
"'.$_POST['nom_proprio'].'"';

        // on lance la requête (mysql_query) et on impose un message
d'erreur si la requête ne se passe pas bien (or die)
        $req = mysql_query($sql) or die('Erreur SQL !<br />'.$sql.'<br
/>'.mysql_error());

        // on récupère le résultat sous forme d'un tableau
        $data = mysql_fetch_array($req);

        // on libère l'espace mémoire alloué pour cette interrogation de
la base
        mysql_free_result ($req);
        mysql_close ();

// on affiche le résultat
        echo 'Le numéro de téléphone est : '.$data['telephone'];
}
else {
        echo 'La variable nom_proprio n\'est pas déclarée';
}
?>
</body>
</html>
```

14. Insérer des données dans votre base

La table liste_proprietaire :

N. du propriétaire	Propriétaire	N. tél
1	LA GLOBULE	06-48-85-20-54
2	Jeremy	06-85-98-78-12
3	Benoît	06-47-01-59-36

La table liste_disque :

N. du propriétaire	Auteur	Titre
1	Cassius	Au rêve
1	Daft Punk	Discovery
2	Cassius	Au rêve
2	Télépopmusik	Genetic world
3	Clamaran	Release yourself
2	Bob Sinclar	Paradise

Supposons alors que l'on décide d'ajouter un nouveau propriétaire de disques : tibo.
Pour insérer ce nouveau propriétaire, il faut fournir au SGBD les informations lui permettant d'insérer ce nouveau tuple dans la table liste_proprietaire. Ces informations sont :

- le numéro du nouveau propriétaire
- le nom du nouveau propriétaire
- son numéro de téléphone

En revanche, il n'est pas nécessaire de fournir au SGBD le numéro du nouveau propriétaire car cet attribut à été déclaré AUTO_INCREMENT lors de la création de la table. Ceci implique que le SGBD sait, lors d'une nouvelle insertion, qu'il faut qu'il prenne dans la table liste_proprietaire le numéro le plus grand et qu'il l'augmente de un, et ce nouveau numéro correspondra au numéro de notre nouveau propriétaire.

On aura alors :

```
INSERT INTO liste_proprietaire VALUES ('','tibo','06-98-42-01-36');
```

Passons tout de suite à l'insertion d'un nouveau tuple, et ce, à partir d'une page WEB.
Supposons que l'on désire insérer exactement le même tuple que dans l'exemple précédent.

On aura alors :

```
<?php
// on se connecte à notre base
$base = mysql_connect ('serveur', 'login', 'pass');
mysql_select_db ('ma_base', $base) ;
?>
<html>
<head>
<title>Insertion de tibo dans la base</title>
</head>
<body>
<?
// lancement de la requete
$sql = 'INSERT INTO liste_proprietaire VALUES ("", "tibo", "06-98-42-01-
36")';

// on insere le tuple (mysql_query) et au cas où, on écrira un petit
message d'erreur si la requête ne se passe pas bien (or die)
mysql_query ($sql) or die ('Erreur SQL !'.$sql.'<br />'.mysql_error());

// on ferme la connexion à la base
mysql_close();
?>
Tibo vient d'être inseré dans la base.
</body>
</html>
```

Imaginons, alors que l'on désire alors ajouter à la base un disque grâce à la contribution de tibo.

Dans l'absolu, réfléchissons une minute sur ce que nous avons besoin :

- un nouveau disque (soit son auteur et son titre).
- le numéro de tibo dans la table liste_proprietaire.

En fait, il faudrait sélectionner le numéro de tibo dans la table_proprietaire
(par le biais d'une requête SQL de type SELECT).

Cependant, imaginons que nous n'avons pas encore inséré le propriétaire tibo dans notre base de données, et que l'on désire directement insérer ce nouveau propriétaire ainsi qu'un disque lui appartenant. Nous allons voir comment récupérer simplement le nouveau numéro qui vient d'être inséré (donc celui de tibo) et ainsi l'utiliser pour insérer notre disque.

```
<?php
// on se connecte à notre base
$base = mysql_connect ('serveur', 'login', 'pass');
mysql_select_db ('ma_base', $base) ;
?>
<html>
<head>
<title>Insertion de tibo et d'un nouveau disque dans la base</title>
</head>
<body>
<?
// on prépare la requête
$sql = 'INSERT INTO liste_proprietaire VALUES("", "tibo", "06-98-42-01-
36")';

// on insère le tuple (mysql_query) et au cas où, on écrira un petit
message d'erreur si la requête ne se passe pas bien (or die)
mysql_query ($sql) or die ('Erreur SQL !'.$sql.'<br />'.mysql_error());

// on récupère le dernier numéro inséré, soit le numéro de tibo
$numero_insere = mysql_insert_id();

// on insère le tuple (mysql_query) et au cas où, on écrira un petit
message d'erreur si la requête ne se passe pas bien (or die)
$sql = 'INSERT INTO liste_disque VALUES ("'.$numero_insere.'", "The
supermen lovers", "The player")';

// on insère le tuple (mysql_query) et au cas où, on écrira un petit
message d'erreur si la requête ne se passe pas bien (or die)
mysql_query ($sql) or die ('Erreur SQL !'.$sql.'<br />'.mysql_error());

// on ferme la connexion à la base
mysql_close();
?>
Tibo vient d'être inseré dans la base, ainsi que son nouveau disque : The
player des Supermen lovers.
</body>
</html>
```

Que faire en plus ?

Et bien, tout comme dans le tutorial précédent, vous pouvez rendre vos insertions vraiment dynamiques en effectuant tout simplement vos insertions à partir des valeurs fournies par un formulaire.

Imaginons que l'on désire insérer des nouveaux disques. Supposons que l'on dispose d'une page html contenant un formulaire permettant de saisir le nom du propriétaire, et que ce formulaire vous demande également le titre d'un album ainsi que son interprète (on suppose également que le champ action de notre formulaire correspond au nom de la page PHP qui traite les données, soit la page contenant le code ci-dessous).

On suppose enfin, que le champ du formulaire contenant le nom du propriétaire porte le nom proprio (on pourra alors utiliser la variable $_POST['proprio'] dans notre page PHP, tout en supposant de notre formulaire à une méthode POST et npn GET), que le champ contenant l'interprète porte le nom interprète et que le champ contenant le titre porte le nom titre.

```php
<?php
// on se connecte à notre base
$base = mysql_connect ('serveur', 'login', 'pass');
mysql_select_db ('ma_base', $base) ;
?>
<html>
<head>
<title>Insertion de nouveaux disques dans la base</title>
</head>
<body>
<?
// on teste si les variables du formulaire sont bien déclarées
if (isset($_POST['proprio']) && isset($_POST['interprete']) &&
isset($_POST['titre'])) {

        // on prépare la requête pour récupérer le numero du propriétaire
        $sql = 'SELECT numero FROM liste_proprietaire WHERE nom =
"'.$_POST['proprio'].'"';

        // on lance la requête (mysql_query) et on impose un message
d'erreur si la requête ne se passe pas bien (or die)
        $req = mysql_query($sql) or die('Erreur SQL !<br />'.$sql.'<br
/>'.mysql_error());

        // on récupère le résultat sous forme d'un tableau
        $data = mysql_fetch_array($req);

        // on libère l'espace mémoire alloué pour cette interrogation de
la base
        mysql_free_result ($req);

        // on insère le tuple (mysql_query) et au cas où, on écrira un
petit message d'erreur si la requête ne se passe pas bien (or die)
        $sql = 'INSERT INTO liste_disque VALUES("'.$data['numero'].'",
"'.$_POST['interprete'].'", "'.$_POST['titre'].'")';

        // on insère le tuple (mysql_query) et au cas où, on écrira un
petit message d'erreur si la requête ne se passe pas bien (or die)
        mysql_query ($sql) or die ('Erreur SQL !'.$sql.'<br
/>'.mysql_error());

        // on ferme la connexion à la base
        mysql_close();

        echo 'Nous venons d\'insérer un nouveau disque :
'.$_POST['titre'].' de '.$_POST['interprete'].' appartenant à
'.$_POST['proprio'];
}
else {
        echo 'Les variables du formulaire ne sont pas déclarées';
}
?>
</body>
</html>
```

15. Modifier des données de votre base

La table liste_proprietaire :

N. du propriétaire	Propriétaire	N. tél
1	LA GLOBULE	06-48-85-20-54
2	Jeremy	06-85-98-78-12
3	Benoît	06-47-01-59-36
4	Tibo	06-98-42-01-36

La table liste_disque :

N. du propriétaire	Auteur	Titre
1	Cassius	Au rêve
1	Daft Punk	Discovery
2	Cassius	Au rêve
2	Télépopmusik	Genetic world
3	Clamaran	Release yourself
2	Bob Sinclar	Paradise
4	The supermen lovers	The player

Supposons que Benoît vienne de changer son numéro de portable (et que son nouveau numéro est : 06-55-99-10-00), il faudra alors faire la modification dans la base de données afin que soit ancien numéro soit remplacé par le nouveau.

On aura alors :

```
UPDATE liste_proprietaire
SET telephone="06-55-99-10-00"
WHERE nom="Benoît";
```

On peut également modifier plusieurs attributs d'un même tuple.
On pourrait alors très bien avoir une table ressemblant à ceci :

N.	Nom	N. tél	Adresse	Age
1	LA GLOBULE	06-48-85-20-54	2, rue des lilas	23
2	Jeremy	06-85-98-78-12	4, rue des fauvettes	22
3	Benoît	06-55-99-10-00	2, rue des tulipes	66
4	Tibo	06-98-42-01-36	8, rue du facteur	23

Supposons que Benoît ait 65 ans au lieu de 66.
Supposons également que l'adresse de Benoît soit erronée et qu'il n'habite pas 2 rue des tulipes, mais 3 rue des tulipes.

On aura alors :

> **UPDATE** liste_proprietaire
> **SET** adresse="3, rue des tulipes", age="65"
> **WHERE** nom="Benoît";

On remarque alors qu'il suffit de séparer les diverses modifications opérées sur un même tuple par une simple virgule.

Voyons maintenant comment effectuer ces modifications dans une page PHP.
Pour ce faire, prenons notre deuxième modification, celle concernant l'adresse et l'age de Benoît.

On aura alors :

```
<?php
// on se connecte à notre base
$base = mysql_connect ('serveur', 'login', 'pass');
mysql_select_db ('ma_base', $base) ;
?>
<html>
<head>
<title>Modification du tél et de l'adresse de Benoît</title>
</head>
<body>
<?
// lancement de la requête
$sql ='UPDATE liste_proprietaire SET adresse="3, rue des tulipes",
age="65" WHERE nom="Benoît"';

// on exécute la requête (mysql_query) et on affiche un message au cas où
la requête ne se passait pas bien (or die)
mysql_query($sql) or die('Erreur SQL !'.$sql.'<br />'.mysql_error());

// on ferme la connexion à la base
mysql_close();
?>
L'adresse et l'age de Benoît viennent d'être modifiés.
</body>
</html>
```

Attention

En effet, les modifications peuvent être vraiment dangereuses pour votre base de données.
En effet, si vous ne prenez pas un minimum de précaution pour effectuer vos modifications, vous pouvez très bien modifier un tuple d'une table et perde en même temps la jointure avec

une autre table.

En effet, nous savons que nos tables liste_disque et liste_proprietaire sont liées par l'intermédiaire du numéro de propriétaire (la jointure).

Ceci implique donc que si pour une raison ou pour une autre, nous sommes amené à modifier ce numéro (dans la table liste_proprietaire par exemple), il faudra également penser à faire la modification de ce même numéro dans l'autre table (liste_disque) afin que la jointure entre les deux tables soit toujours fonctionnelle.

Que faire en plus ?

Vous pouvez effectuer des modifications de tuples par le biais de formulaires.

Par exemple, supposons que l'on dispose d'un page WEB comportant un formulaire disposant des champs suivants :

- un menu déroulant permettant de choisir le nom d'un propriétaire.
- un champs texte classique pour saisir une nouvelle

Supposons ensuite que ce formulaire a pour balise ACTION la page traitement.php qui nous permet de modifier l'adresse du propriétaire en question.

```
<?php
// on se connecte à notre base
$base = mysql_connect ('serveur', 'login', 'pass');
mysql_select_db ('ma_base', $base) ;
?>
<html>
<head>
<title>Modification de l'adresse d'un propriétaire</title>
</head>
<body>
<?
// on teste si les variables du formulaire sont déclarées
if (isset($_POST['nouvelle_adresse']) && isset($_POST['proprio'])) {

        // lancement de la requête
        $sql = 'UPDATE liste_proprietaire SET
adresse="'.$_POST['nouvelle_adresse'].'" WHERE
nom="'.$_POST['proprio'].'"';

        // on exécute la requête (mysql_query) et on affiche un message
au cas où la requête ne se passait pas bien (or die)
        mysql_query($sql) or die('Erreur SQL !'.$sql.'<br
/>'.mysql_error());

        // on ferme la connexion à la base
        mysql_close();

        // un petit message permettant de se rendre compte de la
modification effectuée
        echo 'La nouvelle adresse de '.$_POST['proprio'].' est :
'.$_POST['nouvelle_adresse'];
}
else {
        echo 'Les variables du formulaire ne sont pas déclarées';
}
?>
</body>
</html>
```

14. Supprimer des données de votre base

Après avoir vu l'affichage des données provenant d'une base de données, l'insertion et la modification de ces mêmes données, voyons maintenant la dernière opération fondamentale concernant ces base de données : la suppression de tuples.

Reprenons alors nos deux tables liste_proprietaire et liste_disque que nous avons utilise pendant tous les tutoriaux concernant les bases de données.

Je vous rappelle que l'on avait alors :

La table liste_proprietaire :

N. du propriétaire	**Propriétaire**	**N. tel**
1	LA GLOBULE	06-48-85-20-54
2	Jeremy	06-85-98-78-12
3	Benoît	06-55-99-10-00
4	Tibo	06-98-42-01-36

La table liste_disque :

N. du propriétaire	**Auteur**	**Titre**
1	Cassius	Au rêve
1	Daft Punk	Discovery
2	Cassius	Au rêve
2	Télépopmusik	Genetic world
3	Clamaran	Release yourself
2	Bob Sinclar	Paradise
4	The supermen lovers	The player

Voyons alors, en SQL, comment supprimer un tuple de la table liste_proprietaire.

Supposons que l'on désire supprimer Tibo de notre base de données. On écrira alors :

```
DELETE from liste_proprietaire
WHERE nom="Tibo";
```

Attention

Lorsque l'on effectue une suppression de t-uples, il faut toujours faire attention à effacer non seulement les t-uples de la table dont on veut supprimer le(s) élément(s) mais éventuellement les autres t-uples d'une autre table (si les deux tables sont jointes par le biais d'un attribut).

Voyons maintenant comment effectuer ces suppressions par le biais d'une page WEB écrite en PHP. Prenons par exemple le cas d'une page PHP permettant la suppression de Tibo de la base de données ainsi que de toutes les informations le concernant (c'est-à-dire les disques lui appartenant).

On aura alors :

```
<?php
// on se connecte à notre base
$base = mysql_connect ('serveur', 'login', 'pass');
mysql_select_db ('ma_base', $base) ;
?>
<html>
<head>
<title>Suppression de Tibo de la base</title>
</head>
<body>
<?
// lancement de la requête pour effacer Tibo
$sql ='DELETE from liste_proprietaire WHERE nom="Tibo"';

// on exécute la requête (mysql_query) et on affiche un message au cas où
la requête ne se passait pas bien (or die)
mysql_query($sql) or die('Erreur SQL !'.$sql.'<br />'.mysql_error());

// lancement de la requête pour effacer les disques de Tibo (je vous
rappelle que Tibo à le numéro 4)
$sql ='DELETE from liste_disque WHERE numero="4"';

// on exécute la requête (mysql_query) et on affiche un message au cas où
la requête ne se passait pas bien (or die)
mysql_query($sql) or die('Erreur SQL !'.$sql.'<br />'.mysql_error());

// on ferme la connexion à la base
mysql_close();
?>
Tibo et tous ces disques ont étés supprimés de la base de données.
</body>
</html>
```

Que faire en plus ?

Il serait intéressant de faire une page WEB contenant un formulaire possédant un menu déroulant permettant de choisir le nom du membre à effacer.
On suppose alors que ce menu déroulant à le champ NAME qui prend la valeur proprio, et que le formulaire a son champ ACTION qui prend la valeur traitement.php.
Ceci implique que dans la page traitement.php, on aura une variable $proprio qui contient le nom du propriétaire à supprimer.

On aura alors le code suivant (pour la page traitement.php placée dans le même répertoire que la page WEB contenant le formulaire) :

```php
<?php
// on se connecte à notre base
$base = mysql_connect ('serveur', 'login', 'pass');
mysql_select_db ('ma_base', $base) ;
?>
<html>
<head>
<title>Suppression d'un membre de la base</title>
</head>
<body>
<?
// on teste si la variable du formulaire est bien déclarée
if (isset($_POST['proprio'])) {

        // on recherche le numero du membre à supprimer
        $sql = 'SELECT numero FROM liste_proprietaire WHERE nom =
"'.$_POST[proprio].'"';

        // on lance la requête (mysql_query) et on impose un message
d'erreur si la requête ne se passe pas bien (or die)
        $req = mysql_query($sql) or die('Erreur SQL !<br />'.$sql.'<br
/>'.mysql_error());

        // on recupere le resultat sous forme d'un tableau
        $data = mysql_fetch_array($req);

        // on recupere la valeur qui nous intersse
        $numero_du_proprio = $data['numero'];

        // on libère l'espace mémoire alloué pour cette interrogation de
la base
        mysql_free_result ($req);

        // lancement de la requête pour effacer notre membre
        $sql ='DELETE from liste_proprietaire WHERE
nom="'.$_POST['proprio'].'"';

        // on exécute la requête (mysql_query) et on affiche un message
au cas où la requête ne se passait pas bien (or die)
        mysql_query($sql) or die('Erreur SQL !'.$sql.'<br
/>'.mysql_error());

        // lancement de la requête pour effacer les disques de notre
membre
        $sql ='DELETE from liste_disque WHERE
numero="'.$numero_proprio.'"';

        // on exécute la requête (mysql_query) et on affiche un message
au cas où la requête ne se passait pas bien (or die)
        mysql_query($sql) or die('Erreur SQL !'.$sql.'<br
/>'.mysql_error());

        // on ferme la connexion à la base
        mysql_close();

        // un petit message afin de voir ce qui s'est passé
        echo 'Nous venons de supprimer '.$_POST['proprio'].' de la base
ainsi que tous ces disques';
}
else {
        echo 'La variable de notre formulaire n\'est pas initialisée.';
}
?>
</body>
</html>
```

15. Les cookies

Mettons les choses au clair.
En effet, beaucoup de personnes d'imaginent que les cookies sont des petites bêtes malveillantes alors que c'est totalement faux.

Un cookie est un petit fichier texte (faisant au maximum 65 Ko) stocké sur le disque dur du visiteur du site. Ce fichier texte permet de sauvegarder diverses informations concernant ce visiteur afin de pouvoir les réutiliser ces informations lors de la prochaine visite du visiteur sur ce même site.

Par exemple, on pourrait très bien stocker dans ce cookie le nom du visiteur et par la suite, afficher son nom à chaque fois qu'il se connectera sur le site (ceci bien sur, s'il n'efface pas les cookies de son disque dur).

Cependant, tout cela n'arrive pas par le saint esprit.
En effet, ceci n'est possible que si le visiteur à entré lui-même ses informations dans un formulaire sur le site.

Les cookies sont stockés, selon votre navigateur Internet, à un certain endroit de votre disque dur. Par exemple, avec un système composé de Windows et du navigateur INTERNET EXPLORER (le plus usité).
Dans cette configuration, les cookies sont stockés dans le répertoire C:WindowsTemporary Internet Files comme ci-dessous :

Voyons à présent comment créer de tels cookies, grâce à la fonction **setcookie**().

Soit alors la portion de code suivante :

```php
<?php
// on définit une durée de vie de notre cookie (en secondes), donc un an
dans notre cas
$temps = 365*24*3600;

// on envoie un cookie de nom pseudo portant la valeur LA GLOBULE
setcookie ("pseudo", "LA GLOBULE", time() + $temps);
?>
```

Explications

Grâce à ce code, nous venons d'envoyer, chez le client (donc le visiteur du site) un cookie de nom pseudo portant la valeur LA GLOBULE.
De plus, **time**() retournant le nombre de secondes écoulées depuis le 1er janvier 1970 jusqu'à l'instant présent, nous imposons que le cookie ai une durée de vie de un an (soit en fait l'instant présent plus un an, donc un an).

Enfin, maintenant, si le visiteur ne supprime pas ce cookie, et bien, dans toutes les pages WEB de notre site, on pourra accéder à la variable $pseudo qui contiendra la chaîne de caractères LA GLOBULE.

En revanche, l'envoie d'un cookie ayant la même valeur pour tous les visiteurs d'un site, ce n'est pas vraiment intéressant.

Supposons alors que sur une page de notre site WEB, nous souhaitons faire en sorte que si le visiteur vient pour la première fois (ou qu'il a supprimer ses cookies), et bien, il aurait alors, la possibilité de saisir son nom dans un formulaire, ou bien s'il ne s'agit pas de sa première visite, d'afficher tout simplement Bonjour puis son nom.

On aurait alors le code suivant pour notre page (par exemple index.php) :

```
<html>
<head>
<title>Index du site</title>
<body>

<?
// on teste la déclaration de notre cookie
if (isset($_COOKIE['pseudo'])) {

        echo 'Bonjour '.$_COOKIE['pseudo'].' !';

        // si le cookie n'existe pas, on affiche un formulaire permettant
au visiteur de saisir son nom
        echo '<form action="./traitement.php" method="post">';
        echo 'Votre nom : <input type = "texte" name = "nom"><br />';
        echo '<input type = "submit" value = "Envoyer">';

}
else {
        echo 'Notre cookie n\'est pas déclaré.';
}
?>

</body>
</html>
```

Et le code pour la page traitement.php :

```
<?php
If (isset($_POST['nom'])) {
        // on définit une durée de vie de notre cookie (en secondes),
donc un an dans notre cas
        $temps = 365*24*3600;

        // on envoie un cookie de nom pseudo portant la valeur de la
variable $nom, c'est-à-dire la valeur qu'a saisi la personne qui a rempli
le formulaire
        setcookie ("pseudo", $_POST['nom'], time() + $temps);

        // fonction nous permettant de faire des redirections
        function redirection($url){
            if (headers_sent()){
                print('<meta http-equiv="refresh"
content="0;URL='.$url.'">');
            }
            else {
                header("Location: $url");
            }
        }

        // on effectue une redirection vers la page d'accueil
        redirection ('index.php');
}
else {
        echo 'La variable du formulaire n\'est pas déclarée.';
}
?>
```

Attention

Plusieurs conditions sont à respecter afin que l'utilisation des cookies se passe au mieux :

- l'envoie d'un cookie doit être la première fonction PHP que vous utilisez dans votre script, ce qui veut dire que vous devez utiliser la fonction setcookie() tout en haut de votre script (AUCUN AFFICHAGE ET AUCUN CODE CODE HTML AVANT UN SETCOOKIE). Si d'autres fonctions interviennent avant l'envoie du cookie, celui-ci ne fonctionnera pas.

- Si vous envoyé un cookie sur un poste client celui-ci effacera automatiquement l'ancien cookie qui portait le même nom (si il y en avait un), autrement il le créera.

Note

Pour effacer un cookie, vous devez lancer un cookie qui aura le même nom que le cookie que vous voulez effacer, tout en lui donnant une valeur nulle (vous pouvez également l'envoyer avec un temps de vie dépassé).

16. Les sessions

Pour la sécurité de vos scripts : les sessions.

Afin de transmettre des variables de pages en pages, plusieurs possibilités s'offrent à vous :
- les divers champs des formulaires, qu'ils soient hidden ou non.
- passer les variables directement à travers les liens.
- utiliser les cookies.
- utiliser les sessions.

Cependant, toutes ces possibilités n'offrent pas le même niveau de sécurité.

En effet, certaines de ces possibilités sont vraiment pratiquent dans leurs modes d'utilisation (comme les cookies par exemple mais tout le monde n'est pas obligé d'accepter les cookies), ce qui implique, dans la majorité des cas, un bas niveau de sécurité (cas du passage des variables par les liens, ce qui implique que les variables seront visibles de tout le monde).

De même, faire dans chaque page un formulaire contenant des champs hidden permettant de faire circuler les différentes variables à travers toutes les pages du site n'est pas vraiment pratique.

C'est pourquoi, dans tous ces cas où la sécurité de vos données est primordiale, vous devrez utiliser les sessions qui vous permettront de faire circuler différentes variables (comme un mot de passe par exemple) à travers les pages de votre site, tout en étant assez confortables à l'emploi.

Pour utiliser les sessions, différentes fonctions PHP s'offrent à nous. Voici déjà un petit tableau vous permettant de vous familiariser avec ces différentes fonctions (que nous détaillerons bien sur dans la suite de ce tutorial) :

Fonction	Signification
session_start	Démarre une session
session_register	Enregistre une variable de session
session_unregister	Efface une variable de session
session_is_registered	Vérifie si une variable est déclarée pour la session en cours
session_id	Retourne l'id de la session en cours
session_name	Retourne le nom de la session en cours
session_unset	Detruit toutes les variables de la session en cours
session_destroy	Destruit la session en cours

Sachez également qu'ils existent d'autres fonctions agissant sur les sessions.
En revanche, étant donné qu'elles ne sont pas nécessaires à la compréhension de notre tutorial, nous ne les détaillerons pas ici (cf. la documentation pour de plus amples informations).
Cependant, ces fonctions commencent toujours par session.

Attention

Vous devez savoir que les sessions ne sont accessibles qu'à partir de PHP 4.
Cependant, la plupart des hébergeurs sont aujourd'hui fait évoluer leur PHP en PHP 4.

Afin de voir concrètement comment fonctionnent les sessions, prenons alors un exemple simple. Imaginons que notre site possède une section membre où chaque membre devra se logué avant de pouvoir y entrer. De plus, on aimerait bien être sûr qu'il s'agisse toujours de ce même membre qui est connecté.

On aura alors une page contenant un formulaire permettant à notre visiteur de se connecter à une section membre (page index.htm) :

```
<html>
<head>
<title>Formulaire d'identification</title>
</head>

<body>
<form action="login.php" method="post">
Votre login : <input type="text" name="login">
<br />
Votre mot de passé : <input type="password" name="pwd"><br />
<input type="submit" value="Connexion">
</form>

</body>
</html>
```

D'après cette page, vous pouvez remarquer que lorsque le visiteur le remplira et qu'il cliquera sur le bouton de connexion, on se retrouvera au niveau de la page login.php avec une variable $pseudo qui contiendra le login de notre visiteur ainsi qu'une variable $pwd contenant son mot de passe ; variables qu'il faudra naturellement tester avant de démarrer notre session (car seuls les membres pourront accéder à notre espace membre, espace où l'on utilisera notre session).

On aura alors par exemple (page login.php) :

```php
<?php
// On définit un login et un mot de passe de base pour tester notre
exemple. Cependant, vous pouvez très bien interroger votre base de
données afin de savoir si le visiteur qui se connecte est bien membre de
votre site
$login_valide = "moi";
$pwd_valide = "lemien";

// on teste si nos variables sont définies
if (isset($_POST['login']) && isset($_POST['pwd'])) {

        // on vérifie les informations du formulaire, à savoir si le
pseudo saisi est bien un pseudo autorisé, de même pour le mot de passe
        if ($login_valide == $_POST['login'] && $pwd_valide ==
$_POST['pwd']) {
                // dans ce cas, tout est ok, on peut démarrer notre
session

                // on la démarre :)
                session_start ();
                // on enregistre les paramètres de notre visiteur comme
variables de session ($login et $pwd) (notez bien que l'on utilise pas le
$ pour enregistrer ces variables)
                $_SESSION['login'] = $_POST['login'];
                $_SESSION['pwd'] = $_POST['pwd'];

                // on redirige notre visiteur vers une page de notre
section membre
                header ('location: page_membre.php');
        }
        else {
            // Le visiteur n'a pas été reconnu comme étant membre de
notre site. On utilise alors un petit javascript lui signalant ce fait
            echo '<body onLoad="alert(\'Membre non reconnu...\')">';
            // puis on le redirige vers la page d'accueil
            echo '<meta http-equiv="refresh" content="0;URL=index.htm">';
        }
}
else {
        echo 'Les variables du formulaire ne sont pas déclarées.';
}
?>
```

Remarquer également que nous utilisons notre session_start avant tout code HTML.

Voyons alors le code de la page de notre section membre, la page page_membre.php.
On a :

```php
<?php
// On démarre la session (ceci est indispensable dans toutes les pages de
notre section membre)
session_start ();

// On récupère nos variables de session
if (isset($_SESSION['login']) && isset($_SESSION['pwd'])) {

        // On teste pour voir si nos variables ont bien été enregistrées
        echo '<html>';
        echo '<head>';
        echo '<title>Page de notre section membre</title>';
        echo '</head>';

        echo '<body>';
        echo 'Votre login est '.$_SESSION['login'].' et votre mot de
passe est '.$_SESSION['pwd'].'.';
        echo '<br />';

        // On affiche un lien pour fermer notre session
        echo '<a href="./logout.php">Déconnection</a>';
}
else {
        echo 'Les variables ne sont pas déclarées.';
}
?>
```

Voyons alors le code de la page permettant au membre de se déconnecter (la page logout.php).

```php
<?php
// On démarre la session
session_start ();

// On détruit les variables de notre session
session_unset ();

// On détruit notre session
session_destroy ();

// On redirige le visiteur vers la page d'accueil
header ('location: index.htm');
?>
```

En résumé

- chaque session à un id différent (ce qui permet d'éviter la confusion entre les connexions).

- à chaque page où notre session doit être active, on doit placer un session_start en tout début de page (avant tout code HTML).

- toutes les variables enregistrées au cours de notre session, seront accessibles dans les pages de notre session.

- n'oubliez JAMAIS de détruire vos variables de session lors de la déconnexion.

En respectant ces règles vous pourrez très rapidement faire vous-même votre espace membre, voir même pourquoi pas une boutique en ligne :)

17. Les variables globales

Depuis la version 4.2.0 de PHP, un paramètre de la configuration de PHP (celui concernant les variables globales) est initialisé par défaut à la valeur OFF lors de l'installation (ce qui implique que les variables globales ne sont pas activées) alors qu'auparavant, il était initialisé à ON (là, les variables globales sont activées).

En conséquence, suivant votre hébergeur, et donc de sa configuration de PHP, il se pourrait très bien que tout ce que vous avez vu jusque là ne fonctionne pas sur votre site.

Cependant, cette différence d'initialisation de paramètre n'influence que sur la méthode permettant de récupérer les variables, que ce soit :

- des variables provenant de formulaires POST ou GET
- la valeur des cookies
- des variables de sessions
- des variables d'environnement
- des variables de serveurs

Comme vous le savez, auparavant, pour récupérer nos variables, on les appelait simplement avec un $ suivi de leur nom.

Dans le cas où les variables globales ne sont activées, vous ne pouvez plus utiliser cette méthode afin de récupérer vos variables.

En effet, maintenant, nous récupérerons toutes cas variables et toutes ces valeurs par le biais de tableaux associatifs, que l'on résumer ainsi :

Tableaux associatifs	Description
$_GET	Récupération des variables d'un formulaire GET ou des variables passées par une URL
$_POST	Récupération des variables passées par un formulaire POST
$_FILES	Récupération des variables de fichiers envoyés par un formulaire
$_COOKIE	Récupération des valeurs des cookies
$_SESSION	Récupération des variables de session
$_ENV	Récupération des variables d'environnement
$_SERVER	Récupération des variables serveur

Voyons alors maintenant, cas par cas, comment récupérer ces variables et ces valeurs.

17.1. Le cas des formulaires GET (ou des variables passées par une URL)

Supposons que l'on a une page index.htm contenant un formulaire permettant de saisir un login ainsi qu'un mot de passe.
On pourrait alors très bien avoir une page ressemblant à :

```
<html>
<head>
<title>Formulaire d'identification</title>
</head>

<body>
<form action="login.php" method="get">
Votre identifiant : <input type="text" name="login">
<br />
Votre mot de passe : <input type="password" name="pwd"><br />
<input type="submit" value="Connexion">
</form>

</body>
</html>
```

Pour récupérer nos variables correspondant aux champs login et pwd, nous allons utiliser le tableau associatif $_GET.

On aura lors, par exemple, la page login.php suivante :

```
<html>
<head>
<title>Page de récupération des variables</title>
</head>

<body>
<?
// On teste si nos variables sont déclarées
if (isset($_GET['login']) && isset($_GET['pwd'])) {

        // On fait ce que l'on veut ensuite :)
        echo 'Votre login est '.$_GET['login'].' Et votre mot de passe
est '.$_GET['pwd'];
}
else {
        echo 'Les variables du formulaire ne sont pas déclarées.';
}
?>
</body>
</html>
```

Précisons également, que lorsque l'on passe des variables par une URL, la méthode ne change absolument pas.

En effet, imaginons la page index.htm suivante :

```
<html>
<head>
<title>Juste un lien :)</title>
</head>

<body>
<a href="./login.php?login=GLOBULE&pwd=haha">Notre lien</a>
</body>
</html>
```

Et bien en gardant la même page login.php que précédemment, l'affichage de la page login.php serait strictement identique que dans le cas du formulaire GET.

17.2. Le cas des formulaires POST

La méthode et strictement identique que le cas des formulaires GET, sauf, naturellement, au lieu d'utiliser le tableau associatif $_GET, nous allons ici utiliser le tableau associatif $_POST.

En reprenant les mêmes exemples que précédemment, on aurait alors :

La page index.htm

```
<html>
<head>
<title>Formulaire d'identification</title>
</head>

<body>
<form action="login.php" method="get">
Votre identifiant : <input type="text" name="login">
<br />
Votre mot de passe : <input type="password" name="pwd"><br />
<input type="submit" value="Connexion">
</form>

</body>
</html>
```

La page login.php

```
<html>
<head>
<title>Page de récupération des variables</title>
</head>

<body>
<?
// On teste nos deux variables
if (isset($_POST['login']) && isset($_POST['pwd'])) {

        // On fait ce que l'on veut ensuite :)
        echo 'Votre login est '.$_POST['login'].' Et votre mot de passe
est '.$_POST['pwd'];
}
else {
        echo 'Les variables du formulaire ne sont pas déclarées.';
}
?>
</body>
</html>
```

Récupération d'un fichier par le biais d'un formulaire :

Supposons que l'on dispose d'un formulaire nous permettant d'envoyer un fichier.

```
<html>
<head>
<title>Formulaire permettant d'envoyer un fichier</title>
</head>

<body>
<form action="send_fichier.php" method="post" ENCTYPE="multipart/form-
data">
Votre fichier : <input type="file" name="mon_fichier">
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="20000">
<input type="submit" value="Envoyer">
</form>

</body>
</html>
```

Dans la page send_fichier.php, on a alors accès aux différentes valeurs (suivant les indices) prises par le tableau associatif $_FILES.

Voici les valeurs que contient ce tableau (suivant les différents indices) :

Valeur du tableau	Description
$_FILES['nom_de_la_variable']['name']	Le nom original du fichier qui provient de la machine du client

$_FILES['nom_de_la_variable']['type']	Le type MIME du fichier
$_FILES['nom_de_la_variable']['size']	La taille du fichier en bytes (soit 8 bits ou un octet)
$_FILES['nom_de_la_variable']['tmp_name']	Le nom temporaire du fichier stocké sur le serveur
$_FILES['nom_de_la_variable']['error']	Le code erreur associé à l'upload

Pour afficher ces valeurs, on pourrait très bien le code de la page send_fichier.php qui ressemble à :

```
<html>
<head>
<title>Page de récupération du fichier</title>
</head>

<body>
<?
// On teste les différentes valeurs
if (isset($_FILES['mon_fichier']['name']) &&
isset($_FILES['mon_fichier']['size']) &&
isset($_FILES['mon_fichier']['tmp_name']) &&
isset($_FILES['mon_fichier']['type']) &&
isset($_FILES['mon_fichier']['error'])) {

        // On affiche ces différentes valeurs
        echo  'Nom d\'origine : '.$_FILES['mon_fichier']['name'].'<br
/>';
        echo  'Taille : '.$_FILES['mon_fichier']['size'].'<br />';
        echo  'Nom sur le serveur :
'.$_FILES['mon_fichier']['tmp_name'].'<br />';
        echo  'Type de fichier : '.$_FILES['mon_fichier']['type'].'<br
/>';
        echo  'Code erreur : '.$_FILES['mon_fichier']['error'].'<br />';
}
else {
        echo 'Nos variables ne sont pas déclarées.';
}
?>
</body>
</html>
```

17.2.3. Les cookies

Supposons que nous disposons d'une page send_cookie.php qui nous permet d'envoyer un cookie contenant le nom d'un visiteur.
On aurait alors par exemple :

```php
<?php
$temps = 365*24*5500;
setcookie ("pseudo", "LA GLOBULE", time() + $temps);

header ('Location : index.php');
?>
```

Ensuite, nous allons récupérer la valeur du cookie dans la page index.php, grâce au tableau associatif $_COOKIE.

```php
<html>
<head>
<title>Page de récupération du cookie</title>
</head>

<body>
<?
// On teste notre cookie
if (isset($_COOKIE['pseudo'])) {
        echo 'Votre nom est '.$_COOKIE['pseudo'];
}
else {
        echo 'Cookie non déclaré';
}
?>
</body>
</html>
```

17.2.4. Les variables de sessions

Supposons que l'on a le même formulaire d'identification que dans le tutorial consacré aux sessions.
On aura alors :

La page index.htm

```
<html>
<head>
<title>Formulaire d'identification</title>
</head>

<body>
<form action="login.php" method="post">
Votre login : <input type="text" name="login">
<br />
Votre mot de passé : <input type="password" name="pwd"><br />
<input type="submit" value="Connexion">
</form>

</body>
</html>
```

La page login.php

```
<?php
// On teste nos variables du formulaire d'authentification par le biais
du tableau associatif $_POST
if (isset($_POST['login']) && isset($_POST['pwd'])) {

        // On définit un login et un mot de passe de base pour tester
notre exemple. Cependant, vous pouvez très bien interroger votre base de
données afin de savoir si le visiteur qui se connecte est bien membre de
votre site
        $login_valide = "moi";
        $pwd_valide = "lemien";

        // on vérifie les informations du formulaire, à savoir si le
pseudo saisi est bien un pseudo autorisé, de même pour le mot de passe
        if ($login == $_POST['login'] && $pwd == $_POST['pwd']) {
            // dans ce cas, tout est ok, on peut démarrer notre session

            // on la démarre :)
            session_start ();
            // on enregistre les paramètres de notre visiteur comme
variables de session ($_POST['login'] et $_POST['pwd'])
            $_SESSION['login'] = $_POST['login'];
            $_SESSION['pwd'] = $_POST['pwd'];

            // on redirige notre visiteur vers une page de notre section
membre
            header ('location : page_membre.php');
        }
        else {
            // Le visiteur n'a pas été reconnu comme étant membre de
notre site. On utilise alors un petit javascript lui signalant ce fait
            echo '<body onLoad="alert('Membre non reconnu...')">';
            // puis on le redirige vers la page d'accueil
            header ('location : index.htm');
        }
}
else {
        echo 'Les variables du formulaire ne sont pas déclarées.';
}
?>
```

La page page_membre.php

```
<?php
// On démarre la session (ceci est indispensable dans toutes les pages de
notre section membre)
session_start ();

// On teste pour voir si nos variables ont bien été enregistrées
echo '<html>';
echo '<head>';
echo '<title>Page de notre section member</title>';
echo '</head>';

echo '<body>';

// On teste nos variables de session
if (isset($_SESSION['login']) && isset($_SESSION['pwd'])) {

        // On gère notre affichage
        echo 'Votre login est '.$_SESSION['login'].' et votre mot de
passe est '.$_SESSION['pwd'].'.';
        echo '<br />';

        // On affiche un lien pour fermer notre session
        echo '<a href="./logout.php">Déconnection</a>';
}
else {
        echo 'Les variables de sessions ne sont pas déclarées.';
}
echo '</body>';
echo '</html>';
?>
```

Là enfin, on utilise le tableau associatif $_SESSION pour récupérer nos deux variables de sessions : login et pwd.

17.2.5. Les variables d'environnement

Dans ce cas, rien de plus simple, puisque pour afficher le contenu d'une variable d'environnement, il suffit d'écrire $_ENV puis le nom de la variable que l'on désire afficher entre crochet.

Exemples

```php
<?php
echo 'Système d\'exploitation : '.$_ENV['OS'];
echo '<br /><br />';
echo 'Chemin du profil utilisateur : '.$_ENV['USERPROFILE'];
?>
```

17.2.6. Les variables de serveur

On procède de la même manière que pour les variables d'environnement.

Exemple

```php
<?php
echo 'Chemin du script courant : '.$_SERVER['PHP_SELF'];
echo '<br /><br />';
echo 'Adresse IP du client : '.$_SERVER['REMOTE_ADDR'];
?>
```

Pour récupérer plus rapidement ces variables :

Afin de récupérer facilement et rapidement vos variables, vous pouvez utiliser la fonction extract. En effet, cette fonction exporte vos tableaux associatifs en créant une variable pour chaque indice de vos tableaux.

Par exemple, si on a une page index.htm contenant un formulaire à méthode POST, et que dans votre page login.php (ciblée par le champ ACTION du formulaire) vous traitez ces variables, vous pouvez écrire la ligne suivante (dans la page login.php, en tout début de page)

```php
<?php
extract ($_POST, EXTR, OVERWRITE);
?>
```

Ce qui va créer des variables qui contiendront comme valeur les différentes valeurs du tableau associatif $_POST.
Variables qui auront comme nom, la valeur de chacun des indices du tableau associatif $_POST.

Ceci implique que si l'on a un formulaire de type POST, avec, mettons un champ de name login et un autre de name pwd, et bien au lieu de récupérer à chaque fois la valeur des variables à la main, par le biais du tableau associatif $_POST, cette fonction nous permet d'obtenir directement des variables $login et $pwd (au lieu des valeurs de tableau $_POST['login'] et $_POST['pwd']).

Exemple

La page index.htm

```
<html>
<head>
<title>Formulaire d'identification</title>
</head>

<body>
<form action="login.php" method="post">
Votre identifiant : <input type="text" name="login">
<br />
Votre mot de passe : <input type="password" name="pwd"><br />
<input type="submit" value="Connexion">
</form>

</body>
</html>
```

La page login.php

```
<html>
<head>
<title>Page de récupération des variables</title>
</head>

<body>
<?
// On récupère nos deux variables
extract ($_POST, EXTR, OVERWRITE);

// On fait ce que l'on veut ensuite :)
echo 'Votre login est '.$login.' Et votre mot de passe est '.$pwd;
?>
</body>
</html>
```

En clair, extract à créer :
- une variable $login comprenant la valeur de tableau $_POST['login']
- une variable $pwd comprenant la valeur de tableau $_POST['pwd']

Le second argument de la fonction sert à gérer les collisions de variables.

Voici les valeurs possibles du second argument de extract :

Valeurs	Description
EXTR_OVERWRITE	Ecrase les variables déjà existantes

EXTR_SKIP	N'écrase pas les variables déjà existantes
EXTR_PREFIX_SAME	Si une variable de même nom existe déjà, une nouvelle variable sera créée avec un préfixe donné en troisième argument à extract
EXTR_PREFIX_ALL	Cela crée de nouvelles variables avec le préfixe passé en troisième argument pour toutes les indices du tableau
EXTR_PREFIX_INVALID	Cela crée de nouvelles variables avec le préfixe passé en troisième argument pour les noms de variables invalides (voir le tutorial sur les déclarations de variables)

18. Le débuggage

Maintenant que vous êtes assez familier avec le PHP, il vous est sûrement déjà arrivé de rester coince par un bug pendant (et oui, ça arrive ^^) des heures entières.

La partie de débuggage est une phase essentielle de la programmation d'une application, ne serait que pour faire un rapport de tests : tester chaque fonction afin de voir ce qu'il se passe dans tous les cas.
Le débuggage est également nécessaire lorsque l'on bloque sur un truc que l'on ne comprend pas.

Avant de tomber dans un tel cas de bug où l'on ne peut plus avancer, il existe de nombreuses règles à respecter afin de minimiser le risque de bug.

18.1. Aérer le code

Aérer le code est très important.
Cela ne sert strictement à rien de vouloir faire de la compression de code en ne sautant pas de lignes dans son code, et ce, en écrivant des instructions les unes à la suite des autres sans sauter de lignes (si si, j'ai déjà vu de tels cas ^^).

Exemple de code non aéré :

```
<?php
$toto = 3; $titi = 4; $somme = $toto + $titi; echo $somme;
?>
```

Voici ce même exemple, bien aéré :

```
<?php
$toto = 3;
$titi = 4;
$somme = $toto + $titi;

echo $somme;
?>
```

Naturellement, cet exemple peut paraître tout bête, mais lorsque votre code commence à faire des centaines de lignes, le premier exemple (non aéré) devient vite très énervent : vous ne vous retrouvez plus.

18.2. Indenter le code

Bien indenter le code vous permet de voir rapidement de voir un aperçu de la structure de votre code.

L'indentation consiste à placer certains éléments clés du code (comme les accolades par exemple) à un endroit bien précis et de s'y tenir afin de lire aisément votre code.

Il existe plusieurs techniques (toutes défendables) pour indenter un code, mais seules deux techniques sont réellement utilisées.

Une première forme d'indentation, consiste à placer une seule instruction par ligne (la dessus, en général, tout le monde est d'accord ^^), et lorsque l'on place une conditionnelle ou bien une boucle, et bien nous plaçons l'accolade ouvrante en fin de ligne de conditionnelle ou de boucle, puis nous utilisons une tabulation sur les instructions contenues dans cette conditionnelle ou dans cette boucle.
Enfin, l'accolade fermante de notre conditionnelle ou de notre boucle se placera au niveau de la conditionnelle ou de la boucle.

Exemple

```
<?php
$toto = 2;
if ($toto == 2) {
echo '$toto vaut 2';
}
elseif ($toto == 3) {
        echo '$toto vaut 3';
}
else {
        echo '$toto n\'est pas égal à 2 ou 3';
}
?>
```

Pour les partisans de l'autre technique, les grandes lignes restent les mêmes, mais la différence se joue au niveau de l'accolade ouvrante des conditionnelles ou des boucles.
En effet, certains mettent l'accolade ouvrante, non pas à la fin de la ligne de conditionnelle ou de boucle, mais à la ligne suivante, et l'accolade étant au même niveau que la conditionnelle ou de la boucle.

En reprenant l'exemple précèdent, nous aurons donc :

```php
<?php
$toto = 2;
if ($toto == 2)
{
echo '$toto vaut 2';
}
elseif ($toto == 3)
{
        echo '$toto vaut 3';
}
else
{
        echo '$toto n\'est pas égal à 2 ou 3';
}
?>
```

Tout comme pour l'aération du code, vous devez bien vous rendre compte qu'un code bien indenté sera beaucoup plus lisible qu'un code mal indenté où il faut à chaque fois deviner où se terminent les conditionnelles et les boucles.

Alors que là, en regardant votre éditeur de texte, pour voir où se termine une conditionnelle, vous n'avez qu'à suivre des yeux le niveau (de tabulation) de votre conditionnelle et de faire défiler le texte.
Dès que vous tomberez sur une accolade fermante, et bien c'est cette accolade qui représente la fin de votre conditionnelle (vous n'avez pas besoin réfléchir de l'endroit où se trouve l'accolade fermant cette conditionnelle).

18.3. Commenter le code

Commenter votre code !!!
Je me rappelle encore d'un professeur qui me disait qu'un code sans commentaire ne servait strictement à rien.

En effet, sur le moment, on écrit des dizaines de lignes de code (et des fois, vraiment pas évidentes au premier abord) que l'on comprend parfaitement (parce que l'on a l'algorithme en tête).

Mais dans un mois ? Dans un an ?
Seriez vous aussi sûr de comprendre en 2 minutes ce que vous avez écrit quelques mois plus tôt ?
Pas sur...

Sans commentaires, votre code est pauvre.
Imaginons également qu'un autre programmeur lise votre code.
Sera-t-il capable de comprendre le cheminement de votre pensée ?

Pour toutes ces raisons, je vous invite chaudement à commenter votre code.
Et j'en ai même personnellement fait les frais.
Combien de fois je ne me suis jamais demandé ce que j'avais dans la tête le jour où j'ai pondu

ce code x, et ce, même pour des langages où je me sens à l'aise.

Attention aussi à ne pas tomber dans l'excès de commentaires.
En effet, cela ne sert strictement à rien de mettre un commentaire de ce genre :

```
<?php
// on affiche la somme
echo $somme;
?>
```

Il ne faut pas non plus prendre tous les programmeurs (ainsi que vous au passage ^^) pour des cruches :)
La documentation PHP existe. Si la personne qui lit votre code ne connaît pas l'utilité de la fonction echo, il lui suffit d'ouvrir son manuel PHP et de voir le rôle cette fonction.

Placer des commentaires sur vos fonctions (2 / 3 lignes de commentaires avant le code de la fonction décrivant les paramètres de la fonction et son rôle ne peut être qu'utile), sur vos sections critiques dans votre code (par exemple sur une difficulté algorithmique), dans vos entêtes de classes pour décrire le rôle de votre classe, le genre d'objets qu'elle génère.

18.4. Comment débugger

Si, malgré toutes les précautions que nous avons vu précédemment, vous rester bloquer avec un script récalcitrant qui ne fonctionne pas comme vous le souhaitez (alors qu'il le devrait selon vous ^^), il va falloir débugger.

Pour débugger, il faut déjà notamment retirer tout ce qui est inutile au fonctionnement du script.

Nettoyer notamment le code PHP de tous ses echo de code html (laisser juste des
histoire d'y voir un peu clair quand même ^^).
En effet, pour le moment, votre script bug. Taper directement dans le vif.
Pour le design, on verra après.

D'ailleurs, pour éviter d'avoir ce genre de problème, je vous conseille de faire tous vos scripts sans aucun artifice de design (une fois que votre script fonctionnera sans problème, vous pourrez alors vous occuper de sa mise en page).

Toujours dans le but de débugger, prenez l'habitude lors de la phase de conception d'un script, d'afficher le contenu de vos variables (afin de bien voir ce qu'elles ont dans le ventre lors de l'exécution du script).

Une petite astuce pour afficher tout ce qui est variable de type chaîne de caractères.
Lorsque vous voulez afficher leur contenu, afficher leur contenu entre deux points par exemple (afin de voir si la variable ne contient pas en début ou en fin de chaîne un espace qui peut être source de bug).

```
<?php
$chaine = " test";
echo '.'.$chaine.'.';
?>
```

En ce qui concerne les variables de type tableau (array), vous pouvez visualiser leur contenu à l'aide de la fonction print_r.

Exemple

```
<?php
$tablo = array ('a' => 'pomme', 'b' => 'banane', 'c' => array ('x', 'y',
'z'));
print_r ($tablo);
?>
```

Ce qui affichera :

Array
(
[a] => pomme
[b] => banane
[c] => Array
(
[0] => x
[1] => y
[2] => z
)
)

Un conseil, lorsque vous faites un print_r, visualiser le en affichant la source de votre page (sous Internet Explorer : Menu affichage Source). Vous verrez ainsi le contenu de votre tableau tout indenté ce qui est beaucoup plus lisible.

Remarque

print_r peut être utilisé sur tous vos types de variables.

18.4.1. Cas des conditionnelles

Il peut arriver que les conditionnelles n'aient pas le comportement souhaité à l'origine.
En effet, quelques fois, votre script ne rentre pas dans le if mais dans le else alors qu'il devrait faire le contraire.
Prenez la même méthode que précédemment en affichant le contenu de vos variables afin de voir ce qui cloche.

Prenez également attention aux tests de votre conditionnelle.

En effet, en écrivant par exemple :

```
<?php
$toto = 5;
if ($toto = 4) {
        echo '$toto vaut 4';
}
?>
```

Et bien votre code passera toujours dans le if, et il affichera toujours $toto vaut 4.
Ceci est du à l'utilisation d'un seul = pour faire votre test (vous faites en fait une affectation au lieu d'une comparaison qui elle se fait avec ==).

18.4.2. Cas des boucles

Le cas des boucles est plus ou moins similaire à celui des conditionnelles.

En effet, si votre boucle ne démarre pas du tout, vérifier la valeur de votre compteur à l'initialisation de la boucle.
De même, si votre boucle semble tourner à l'infini, vérifier bien que la valeur pour la sortie de la boucle arrivera à coup sur.

Si par contre votre boucle effectue des traitements non voulus sur vos données, prenez l'habitude de placer un echo dans votre boucle afin de voir la valeur de vos variables à chaque passage de boucle.
Pour pourrez ainsi mieux apprécier le comportement de votre boucle sur vos variables.

Exemple

```
<?php
$toto = 2;
for ($i = 0; $i < 5; $i++) {
        $resultat = $toto * $i;
        echo 'Passage numéro '.$i.' => multiplication = '.$resultat;
        echo '<br />';
}
?>
```

18.4.3. Cas des fichiers

Les erreurs arrivent assez facilement avec l'utilisation des fichiers si l'on ne prend pas garde à certains points.

Lorsque vous avez des erreurs en utilisant des fichiers, vérifier toujours :
- d'une part le chemin pour accéder à votre fichier (chemins relatifs / absolus)
- d'autre part le chmod de ce fichier (afin de voir si vous avez les droits pour accéder à ce fichier).

18.4.4. Problèmes avec MySQL

Voici plusieurs conseils qui vous permettront d'éviter certaines erreurs incompréhensibles avec MySQL.

Tout d'abord, prenez l'habitude de placer vos requêtes SQL dans des variables.

Cela peut paraître rédhibitoire au départ, mais cela a plusieurs avantages.
En effet, en plaçant vos requêtes SQL dans une variable (par exemple $sql, au lieu de faire directement un mysql_query) vous allez pouvoir afficher votre requête SQL (via un echo $sql), ce qui constitue un réel avantage dans le cas de requêtes contenant des variables gérées par PHP (cela vous permet de bien voir si la requête contient les bonnes valeurs pour chacun des éléments gérés par PHP).

Un autre avantage découle aussi de ce premier conseil.

En effet, si votre requête à l'air de bien passer mais que, a priori, la récupération des éléments de la requête pose problème, il arrive souvent que l'on se demande si c'est la requête qui s'est bien déroulée et qui ne retourne aucun résultat ou bien si c'est notre code de récupération qui pose problème.

Afin d'en avoir le coeur net, faites un echo de votre $sql, et copier coller votre requête dans votre PHPMyAdmin : si PHPMyAdmin sort bien un résultat, c'est que votre code de récupération n'est pas fameux.

En revanche, si votre PHPmyAdmin ne retourne rien, c'est bien que requête ne retourne aucun résultat.

De même, en avançant toujours dans cette direction, compter toujours le nombre de résultat retourné de votre requête SQL à l'aide d'un mysql_num_rows, ce qui va vous permettre d'afficher un texte au lieu de ne rien avoir sur l'écran et de ne pas comprendre pourquoi il n'y a rien sur l'écran.

Exemple

```php
<?php
$sql = 'SELECT toto FROM table WHERE test="ok"';
$req = mysql_query($sql) or die('Erreur SQL !<br />'.$sql.'<br
/>'.mysql_error());
$nb = mysql_num_rows ($req);

if ($nb == 0) {
        echo 'Aucun résultat retourné.';
}
else {
        // Récupération des résultats et affichage
}
mysql_free_result ($req);
?>
```

Prenez également l'habitude de mettre un or die muni de la fonction mysql_error sur vos lancement de requêtes SQL afin de voir (si la requête ne passe pas) ce qui pose problème.

Faites aussi toujours un mysql_free_result sur votre requête de type SELECT lorsque celle-ci est terminée afin de libérer la mémoire nécessaire à l'exécution de votre requête.

En effet, d'une part, cela soulage le serveur, et d'autre part, cela évite de récupérer les résultats d'une autre requête faite précédemment.

18.4.5. Les messages d'erreurs fréquents

Enfin, si malgré toutes ces précautions, il vous arrive de bloquer sur une erreur, voici un tableau regroupant les erreurs les plus communes que l'on peut avoir en programmant avec PHP accompagnées de petits indices vous permettant de les résoudre.

Erreur	Remède
Parse error: parse error in xxxx.php on line y	Il s'agit d'une erreur de syntaxe. Vérifiez si vous n'avez pas oublié un ; marquant la fin d'une instruction. Verifier également si il ne manque pas un $ (dollar) devant le nom d'une variable. N'hésitez pas à contrôler les lignes précédentes. L'erreur se trouve souvent juste au-dessus.
Warning: php_SetCookie called after header has been sent in xxxx.php on line y	Vous avez tenté d'initialiser un cookie après que l'entête HTTP soit envoyé au client. Vérifiez si une sortie (echo, print, message d'erreur, ligne blanche, code html avant les tags php) ne se fait pas avant votre initialisation de cookie
Warning: MySQL Connection Failed: Access denied for user:	Erreur de connexion à la base MySQL. Vérifiez vos paramètres de connexion
Warning: Unable to create [chemin] No such file or directory in your script on line [numero]	Le chemin vers le répertoire sensé contenir le fichier ou bien le chemin du répertoire dans lequel le fichier doit être crée est incorrect

Warning: 0 is not a MySQL result index in xxxx.php on line y	Erreur probable au niveau de la requête SQL. Vérifiez votre requête SQL : en particulier les champs manipulés, le nom de ou des tables impliquées, etc...
Warning: Variable $zzzz is not an array or string in xxxx.php on line y	Vous tentez de manipuler une valeur numérique avec une fonction dédiée aux chaînes ou aux tableaux.
Warning: Variable $zzzz is not an array or object in xxxx.php on line y	Vous tentez de manipuler une valeur numérique avec une fonction dédiée aux tableaux ou aux objets.
Warning: Cannot add header information headers already sent in xxxx.php on line y	Vous avez tenté d'effectuer un Header après que l'entête HTTP ait envoyé au client. Vérifiez si une sortie (echo, print, message d'erreur, voir même du code html) ne s'exécute pas avant votre Header
Fatal error: Maximum execution time exceeded in xxxx.php on line y	PHP dispose d'un mécanisme permettant de se prémunir des scripts susceptibles d'engendrer un temps d'exécution trop important pouvant saturer un serveur. Par défaut, ce temps est de 30 secondes.
Fatal error: Allowed memory size of 8388608 bytes exhausted (tried to allocate x bytes) in yyyy.php on line z	PHP dispose d'un mécanisme permettant de se prémunir des scripts susceptibles d'engendrer une consommation mémoire trop importante pouvant saturer un serveur. Par défaut, une limite est fixée à environ 8 Mo (8388608 octets).
Fatal Error: Call to undefined function: xxxx() in yyy.php on line z	La fonction que vous appelez n'existe pas. Ce peut-être une fonction liée à une librairie externe (GD, Zlib, PDF, etc.). Dans ce cas, un simple phpinfo() vous renseignera sur les paramètres de compilation de votre version de PHP. Peut-être s'agit-il sinon d'une de vos propres fonctions. Vérifiez alors qu'elle existe (notamment si votre script y accéde bien si elle se trouve dans un autre fichier). Et dans tous les cas, contrôlez de plus près le nom de la fonction appelée (orthographe, etc.). Une erreur de frappe est vite arrivée.
Fatal Error: Cannot redeclare xxxx() in yyy.php on line z	Vous avez certainement déclaré plusieurs fois la même fonction. Contrôlez à nouveau l'ensemble des fonctions que vous avez créées. Et n'hésitez pas à vérifier également dans les éventuels fichiers inclus. C'est souvent dans un script secondaire que vous trouverez le doublon. Veillez aussi à ne pas utiliser le nom d'une fonction propre à PHP ou à l'une de ses librairies.
Fatal error: Input in flex scanner failed in xxxx on line y	Vérifiez vos include et require. Il y a fort à croire que vous avez indiqué un chemin incomplet (genre /usr/local/ sans préciser de fichier).
Failed opening '%s' for inclusion (include_path='%s')	Le fichier n'a pas pu être inclus dans votre script, car PHP n'a pas pu y accéder : vérifiez les droits (utilisateur PHP, droits du fichier), les noms et chemins du fichier inclus.
file("%s") - Bad file descriptor	Problème d'accès à un fichier avec la fonction file(). Vérifiez bien que l'URL est valide. (l'URL "http://www.super.php") est invalide alors qu'une erreur de type 404 sera valide.
Wrong parameter count for	La fonction est appelée avec un nombre insuffisant de

%s()	paramètre, ou bien avec trop de paramètres. Certaines fonctions ont besoin d'un minimum de paramètres (array()), et généralement d'un maximum.
stat failed for %s (errno=%d - %s)	Impossible d'accéder au fichier (problème de droits ou de chemin d'accès).

19. La portée des variables

Voyons maintenant un caractère propre à beaucoup de langages de programmation fonctionnels, dits, de haut niveaux : la portée des variables.

En PHP, jusqu'à présent, lorsque vous déclariez une variable dans votre script, vous aviez l'habitude d'accéder à cette variable dans tout le reste de votre page.

En effet, d'après le code suivant :

```
<?php
$toto = 5;
// fin du script
?>
```

$toto est une variable qui sera accessible par votre script PHP une fois sa déclaration faite, c'est-à-dire juste après l'instruction $toto = 5.

On peut observer le même comportement lorsque l'on utilise des include.

En effet, une fois que l'on a déclaré une variable, cette variable est accessible directement dans tous les scripts que l'on inclut à notre script courant.

Exemple

```
<?php
$toto = 5;
include ('script.php');
?>
```

Ici, $toto sera accessible, c'est à dire que l'on pourra l'appeler directement (elle contiendra alors sa valeur : 5) dans le script script.php.

Vous savez également, qu'une variable déclarée dans le corps d'une fonction n'est accessible que dans le corps de cette même fonction.

Exemple

```
<?php
$toto = 5;
function ma_fonction () {
        echo $toto;
}

ma_fonction();
?>
```

Dans ce cas, l'exécution de ce code PHP n'affichera rien vu que $toto (contenu dans le code de la fonction ma_fonction()) n'a strictement aucun rapport avec la variable $toto contenu dans le script courant.

On appellera la variable $toto (celle contenue dans le code de la fonction) comme étant une variable locale à la fonction ma_fonction() (elle n'est pas globale au script).

19.1. Le mot clé global

Sachez qu'il est possible d'utiliser dans le code même de vos fonctions des variables que vous avez déclarées dans votre script courant à l'aide du mot clé global.

Reprenons notre exemple précédent :

```
<?php
$toto = 5;
function ma_fonction () {
        global $toto;
        echo $toto;
}

ma_fonction();
?>
```

Cet exemple affichera :

```
5
```

En effet, dans ce script, nous déclarons la variable (qui était jusque là locale) $toto comme étant une variable globale du script.
PHP sait alors qu'il doit récupérer la valeur de cette variable dans le script courant.

Il n'y a aucune limite au nombre de variables globales qui peuvent être manipulées par une fonction.

Vous pouvez également effectuer la même opération en utilisant le tableau associatif $_GLOBALS.

Le code suivant donnera le même résultat que le code précédent :

```
<?php
$toto = 5;
function ma_fonction () {
        echo $_GLOBALS["toto"];
}

ma_fonction();
?>
```

Comme vous le voyez, le tableau $GLOBALS est un tableau associatif avec le nom des variables globales comme clef et les valeurs des éléments du tableau comme valeur des variables, ce qui ressemble un peu aux tableau associatifs que vous connaissez déjà comme $_POST ou $_GET qui vous permettent de récupérer la valeur des champs de vos formulaires.

19.2. Le mot clé static

Les amateurs de programmation par objets doivent bien connaître ce terme de static :)

Une variable dite static est une variable locale à une fonction mais qui a la particularité de se souvenir de sa valeur.

Prenons comme exemple le code suivant :

```
<?php
function ma_fonction () {
        $toto = 1;
        echo $toto;
        $toto++;
}
?>
```

Si je lance cette fonction 50 fois, vous allez tous penser que cette fonction affichera 50 fois la valeur 1, et vous aurez raison :)

Dans ce cas, l'incrémentation ne sert à rien puisque lorsque l'on a fini d'exécuter cette fonction PHP "perd connaissance" de la valeur de la variable $toto.

En revanche, si on déclare la variable $toto comme étant une variable statique (grâce au mot clé static), le comportement de cette fonction sera totalement différent.

En effet, vu que $toto est statique, PHP se souviendra de sa valeur lors de la dernière exécution de la fonction.

Avec le code suivant :

```
<?php
function ma_fonction () {
        static $toto = 1;
        echo $toto;
        $toto++;
}
?>
```

Si je lance 50 fois ma fonction, et bien nous verrons sur l'écran, la suite des nombres entre 1 et 50.

Explication : à chaque appel, PHP se souvient de la précédente valeur de la variable $toto (parce qu'elle est static) et notre fonction incrémente alors cette valeur pour ensuite l'afficher.

19.3. Le passage par référence

Dans nos exemples précédents, vous avez que grâce au mot clé static, les fonctions pouvaient récupérer les valeurs des variables globales.

Sachez qu'il est également possible de modifier la valeur d'une variable d'un script (une variable globale) grâce à une fonction en utilisant un passage de variable par référence.
Le passage par référence se caractérise par l'utilisation d'un & avant le nom de votre variable dans les arguments de votre fonction.

Exemple

```
<?php
$toto = 5;
function ma_fonction ($var) {
        $var++;
}
ma_fonction(&$toto);
echo $toto;
?>
```

Ce script affichera :

6

Pour les personnes qui ont fait du C, le passage par référence s'assimile à la notion de pointeur.

En passant une variable par référence à une fonction, nous ne passons pas en fait la variable en elle-même (la preuve : notre fonction utilise une variable $var alors qu'elle modifie tout de même la valeur de la variable $toto du script courant) mais une référence vers la zone

mémoire où est stockée la valeur de notre variable $toto.

Notre fonction modifie alors directement cette zone mémoire en lui donnant une nouvelle valeur, ce qui explique que dans le script courant, et bien la valeur de $toto a changé.

20. La librairie GD

Nous allons donc voir dans ce tutorial comment faire des images dynamiques (c'est-à-dire qui changent suivant certains paramètres) grâce à PHP, qui les générera.

La librairie GD peut être schématisée par "un fichier" qui comporte de nombreuses fonctions permettant de travailler les images. Grâce à cette librairie, vous allez pouvoir modifier vos images, récupérer des informations sur ces images, voir même, créer vous-même vos propres images.

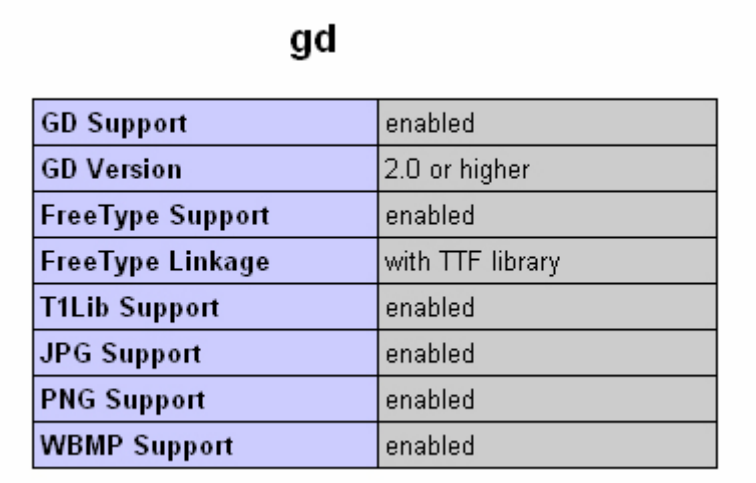
Avant de nous lancer dans la création de superbes images, une petite vérification s'impose. En effet, si la libraire GD n'est pas installée sur votre serveur (votre hébergeur) cela ne sert à rien de poursuivre ce tutorial, vu que les fonctions que nous allons utiliser ne seront pas activées.

Pour savoir si la librairie GD est installée sur votre serveur, vous devez faire un phpinfo. Pour cela, créer un fichier, par exemple phpinfo.php, dans lequel vous allez placer le code suivant :

```
<?php
phpinfo();
?>
```

Ensuite, dans votre navigateur favori, affichez la page ainsi créée.

Faites défiler cette page jusqu'au moment où vous voyez quelque chose qui ressemble à ceci :



gd

GD Support	enabled
GD Version	2.0 or higher
FreeType Support	enabled
FreeType Linkage	with TTF library
T1Lib Support	enabled
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

Là, si comme ici, le GD Support est enabled, c'est tout bon, vous pouvez continuer la lecture de ce tutorial.
Enfin sachez également, que depuis la version 2.0 de la librairie, le format GIF n'est plus supporté, et il a été avantageusement remplacé par le format PNG.

Pour ceux qui n'ont pas la librairie d'installée, le seul remède est de demander gentiment à leur

hébergeur de l'installer :)

Passons de suite à la création de notre première image dynamique.

Prenons par exemple le code suivant, très basique, permettant l'affichage d'une image.
On a par exemple une page que l'on nommera index.php, et qui contiendra :

```
<html>
<head>
<title>Notre page de test</title>
</head>

<body>
<img src="./mon_image.php">
</body>

</html>
```

Et vous notez que bien le lien vers notre image désigne un fichier PHP. C'est tout à fait normal, vu que c'est PHP qui va générer notre image.

Voyons à présent le code de la page mon_image.php.
Imaginons que l'on désire dessiner un rectangle rouge (en fait nous n'allons pas dessiner un rectangle, nous allons en fait créer une image vide remplie avec de la couleur rouge).
On aura alors (pour la page mon_image.php), le code suivant :

```
<?php
// on spécifie le type de document que l'on va créer (ici une image au
format PNG
header ("Content-type: image/png");

// on dessine une image vide de 200 pixels sur 100
$image = @ImageCreate (200, 100) or die ("Erreur lors de la création de
l'image");

// on applique à cette image une couleur de fond, les couleurs étant au
format RVB, on aura donc ici une couleur rouge
$couleur_fond = ImageColorAllocate ($image, 255, 0, 0);

// on dessine notre image PNG
ImagePng ($image);
?>
```

Et voila notre première image générée par PHP.

Cependant, quelques explications sont nécessaires et extrêmement importantes :

- lorsque l'on crée une page WEB avec PHP, on n'est pas obligé de dire à PHP que l'on crée une page WEB, en revanche, lorsque l'on crée une image, on est obligé de le spécifier avec l'instruction :

  ```
  header ("Content-type: image/png");
  ```

- naturellement, si l'on désire créer une image de type JPG, on prendra soin d'écrire :

  ```
  header ("Content-type: image/jpeg");
  ```

- l'appel à la fonction ImageCreate nous retourne en fait une ressource ($image) correspondant à l'image que nous sommes en train de créer.

- nous faisons ensuite appel à la fonction ImageColorAllocate qui a deux fonctions principales : premièrement, elle crée une couleur stockée dans une variable pouvant être réutilisée ultérieurement (ici $couleur_fond), et deuxièmement, elle enregistre cette couleur dans la palette de l'image $image.

Notez aussi, et c'est extrêmement important, que cette couleur étant la première couleur enregistrée dans la palette, elle correspondra en fait à la couleur de fond de notre image (en fait, la première couleur enregistrée dans la palette de couleur correspond à la couleur de fond de l'image).

- on utilise alors la fonction ImagePng pour afficher notre ressource, soit donc notre image ($image).

- notez également que si nous avions créé une image de type JPG, nous aurions utilisé ici l'instruction ImageJpeg.

Après ce premier exemple tout simple, vous devez sûrement vous dire qu'il n'y a rien de dynamique dans tout cela, et que votre Photoshop ferait bien mieux.
A première vue, oui.
Mais vu que notre image est générée par PHP, nous pouvons lui fournir des paramètres. Et l'avantage, il est là. C'est-à-dire que vous allez pouvoir créer des images différentes en fonction de certains paramètres.

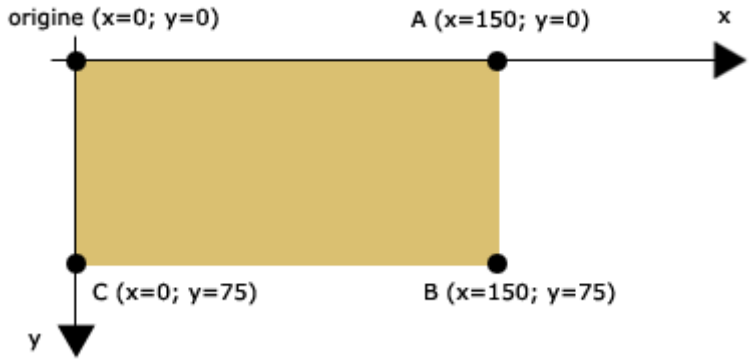
Reprenons alors le code de notre index.php, auquel nous allons greffer un petit formulaire permettant de choisir la couleur de notre image.

On aura alors, par exemple (pour la page index.php) :

```php
<?php
// on teste nos 3 variables pour nos couleurs
if (isset($_POST['rouge']) && isset($_POST['vert']) &&
isset($_POST['bleu'])) {

        // on spécifie le type de document que l'on va créer (ici une
image au format PNG
        header ("Content-type: image/png");

        // on dessine une image vide de 200 pixels sur 100
        $image = @ImageCreate (200, 100) or die ("Erreur lors de la
création de l'image");

        // on applique à cette image une couleur de fond, les couleurs
étant         au format RVB, on obtiendra ici la couleur que
l'utilisateur aura spécifié en paramètre du formulaire
        $couleur_fond = ImageColorAllocate ($image, $_POST['rouge'],
$_POST['vert'], $_POST['bleu']);

        // on dessine notre image PNG
        ImagePng ($image);
}
else {
        echo 'Les variables du formulaire ne sont pas déclarées.';
}
?>
```

Et prenons par exemple le code suivant pour la page mon_image.php (nous ne ferons pas ici la vérification des champs du formulaire : en effet, on supposera que l'utilisateur saisi bien à chaque fois un nombre entre 0 et 255) :

Maintenant que vous avez vu une première approche de l'utilisation de cette librairie, un petit rappel est nécessaire.
En effet, en règle générale, lorsque l'on dessine en programmation, l'origine du repère représentant la position des pixels de notre image est située dans le coin supérieur gauche de l'image.
Cela diffère donc des repères classiques mathématiques où l'origine du repère est toujours situé dans le coin inférieur gauche.
En effet, on peut schématiser ce repère par le dessin suivant :



Et ceci est très important.
En effet, pratiquement toutes les fonctions de cette librairie vous demanderont les coordonnées de différents points.
Autant se mettre tout de suite en accord avec le système.

Conseil

Vu que l'axe des x est orienté de la "gauche" vers la "droite", lorsque vous allez utiliser des fonctions permettant de dessiner des rectangles, des droites, etc, l'idéal est de toujours fournir en arguments les coordonnées du point le plus à gauche de votre figure, puis les coordonnées du point le plus à droite.

Après cette petite précision, voici un autre exemple intéressant de l'utilisation de la librairie GD.

En effet, vous affichez parfois des images sur vos sites WEB, et beaucoup de vos photos partent dans la nature (pillées la plupart du temps par des visiteurs peu attentionnés).
Ne serait-il pas sympathique de pouvoir protéger vos photos à l'aide d'une petite incrustation permettant de "copyrighter" votre image ?
Et bien c'est tout à fait possible.

En effet, à partir de :

gd

GD Support	enabled
GD Version	2.0 or higher
FreeType Support	enabled
FreeType Linkage	with TTF library
T1Lib Support	enabled
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

et de :

Il est tout à fait possible d'obtenir :

gd © lephpfacile.com

GD Support	enabled
GD Version	2.0 or higher
FreeType Support	enabled
FreeType Linkage	with TTF library
T1Lib Support	enabled
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

Pour ce faire, voyons le code pour notre fichier PHP contenant le code générant cette image "copyrightée" (fichier mon_image.php par exemple) :

```php
<?php
// on spécifie le type de fichier créer (ici une image de type jpeg)
header ("Content-type: image/jpeg");

// on crée deux variables contenant les chemins d'accès à nos deux
fichiers : $fichier_source contenant le lien vers l'image à
"copyrighter", $fichier_copyright contenant le lien vers la petite
vignette contenant le copyright (bien sur, on prendra soin de placer les
images sources dans un répertoire "caché" sinon le copyright ne sert à
rien si les visiteurs ont accès aux images sources)
$fichier_source = "./gd.jpg";
$fichier_copyright = "./copyright.jpg";

// on crée nos deux ressources de type image (par le biais de la fonction
ImageCreateFromJpeg)
$im_source = ImageCreateFromJpeg ($fichier_source);
$im_copyright = ImageCreateFromJpeg ($fichier_copyright);

// on calcule la largeur de l'image qui va être copyrightée
$larg_destination = imagesx ($im_source);

// on calcule la largeur de l'image correspondant à la vignette de
copyright
$larg_copyright = imagesx ($im_copyright);
// on calcule la hauteur de l'image correspondant à la vignette de
copyright
$haut_copyright = imagesy ($im_copyright);

// on calcule la position sur l'axe des abscisses de la vignette
$x_destination_copyright = $larg_destination - $larg_copyright;

// on réalise la superposition, le dernier paramètre étant le degré de
transparence de la vignette (cependant, allez voir la fin de ce même
tutorial pour une définition complète de tous les arguments de cette
fonction)
@imageCopyMerge ($im_source, $im_copyright,
        $x_destination_copyright, 0, 0, 0, $larg_copyright,
        $haut_copyright, 70);

// on affiche notre image copyrightée
Imagejpeg ($im_source);
?>
```

Ensuite, afin de voir le résultat, il suffit d'écrire une page toute simple affichant l'image mon_image.php, avec une insertion d'image du genre (par exemple) :

```
<img src="./mon_image.php">
```

Bien sur, pour rendre ce script vraiment dynamique, il suffit de passer à l'image en paramètre le nom du fichier image à "copyrighter".
En effet, comme dans le premier exemple avec l'image basique, on peut faire un lien ressemblant à :

```
<img src="mon_image.php?fichier=toto.jpg">
```

Et ensuite, dans le script mon_image.php, on récupère la variable $_GET['fichier'] qui sera en fait le nom du fichier de l'image à "copyrighter".

Je vous fais confiance pour cette partie :)

Voyons maintenant un autre exemple de l'utilisation de cette librairie : la création de miniatures.
En effet, vous voulez mettre vos photos en ligne, et jusqu'ici, vous faisiez vos miniatures à la main :
- c'était long
- pas toujours réussi

Et pourtant les miniatures sont extrêmement pratiques pour afficher plus rapidement vos grandes images.
En effet, il est toujours agréable de voir une liste de petites images (les miniatures) munies d'un lien permettant d'afficher ces mêmes images dans leur taille originelle.

Et bien, avec la librairie GD, à partir de cette image, vous allez pouvoir créer une miniature ressemblant à ceci :

Détaillons maintenant le code permettant de créer vos miniatures.
En revanche, ici, à la différence des exemples précédents, nous n'allons pas afficher à la volée une image générée par PHP.
En effet, nous allons en fait écrire un script PHP, qui, avec le lien vers une image de type JPG, va créer une miniature de cette même image qui sera sauvegardée sur votre espace disque.
Le nom de la miniature sera en fait le nom du fichier original, précédé de la mention mini_.

On aura alors le code suivant pour ce script de création de miniatures :

```php
<?php
// on donne à PHP le lien vers notre image à miniaturiser
$Image = "metaeffect_001.jpg";

// on impose la taille de la largeur ou de la hauteur de la photo (le
choix entre la largeur ou la hauteur se fait automatiquement, suivant que
la photo est "horizontale" ou "verticale")
$ratio = 150;

// on crée une ressource représentant en fait l'image à miniaturiser
$src=imagecreatefromjpeg($Image);

// on récupère les paramètres de notre image (getimagesize est une
fonction qui retourne un tableau contenant les paramètres d'une image :
sa largeur, son hauteur, son type, etc...)
$size = getimagesize($Image);

// on test si la largeur de l'image est supérieur à sa longueur
if ($size[0] > $size[1]) {
    // on crée une ressource pour notre miniature
    $im=imagecreate(round(($ratio/$size[1])*$size[0]), $ratio);
    // on place dans la ressource que nous venons de créer une copie de
l'image originelle, redimensionnée et réechantillonée
    imagecopyresampled($im, $src, 0, 0, 0, 0,
round(($ratio/$size[1])*$size[0]),$ratio, $size[0], $size[1]);
}
else {
    // si la largeur est inférieure ou égale à la hauteur, on entre dans
ce cas
    // on crée une ressource pour notre miniature
    $im=imagecreate($ratio, round(($ratio/$size[0])*$size[1]));
    // on place dans la ressource que nous venons de créer une copie de
l'image originelle, redimensionnée et réechantillonée
    imagecopyresampled($im, $src, 0, 0, 0, 0, $ratio,
round($size[1]*($ratio/$size[0])), $size[0], $size[1]);
}

// on définit le nom de notre miniature
$miniature = "mini_$Image";

// on crée notre miniature
ImageJpeg ($im, $miniature);
?>
```

Et surtout, remarquez bien que nous n'avons pas utilisé la fonction header.
En effet, ici, on ne cherche pas à "afficher" une image générée à la volée.
Ce script inspecte en fait une image, et à partir de celle-ci, il en crée une nouvelle.
Pour voir le résultat de cette opération, et donc voir votre miniature, vous devrez créer une page html contenant la balise suivante :

```
<img src="mini_metaeffect_001.jpg ">
```

Enfin, afin d'accroître vos talents de Picasso des temps modernes, voyons un dernier exemple : la création d'un histogramme pour un script de statistiques par exemple.
En effet, bon nombre de scripts de statistiques affichent la popularité de vos pages sur de jolis dessins.

Et bien sachez que c'est possible de faire quelque chose d'équivalent à l'aide de la librairie GD.
Pour ce faire, je vous propose la création d'un histogramme permettant d'afficher des "bâtons" représentants le nombre de pages vues de votre site sur une année (un bâton représentant un mois de l'année).

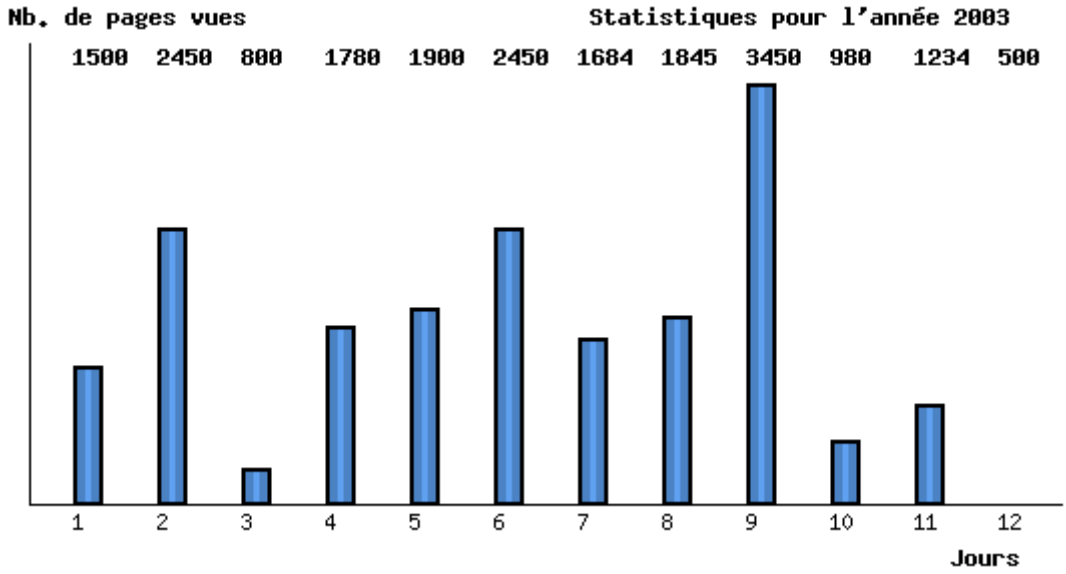
Pour ce faire, voici par exemple le code du script stats_year.php (il s'agira d'une image dynamique, crée à la volée, l'utilisation d'un header sera donc nécessaire) :

```
<?php
// on définit un tableau contenant le nombre de page vues par mois : par
exemple, on suppose que 1500 pages du site ont été vues en janvier, 2450
en février, etc... Bien sur, pour que ce script soit vraiment valides,
vous n'allez pas déclarer ce tableau, car sinon, les bâtons seront
toujours les mêmes :) Vous allez plutôt effectuer une requête SQL sur
votre base de données permettant de récupérer le nombre de pages vues de
votre site par mois. Ensuite, il suffira d'appeler le script avec ces 12
paramètres dans votre page html (en faisant par exemple : <img
src="./images/stats_year.php?visite_par_mois[1]=1500.......">)
$visite_par_mois[1]=1500;
$visite_par_mois[2]=2450;
$visite_par_mois[3]=800;
$visite_par_mois[4]=1780;
$visite_par_mois[5]=1900;
$visite_par_mois[6]=2450;
$visite_par_mois[7]=1684;
$visite_par_mois[8]=1845;
$visite_par_mois[9]=3450;
$visite_par_mois[10]=980;
$visite_par_mois[11]=1234;
$visite_par_mois[12]=500;

// on calcule le nombre de pages vues sur l'année
$max_visite = max($visite_par_mois);

// on spécifie le type d'image que l'on va créer, ici ce sera une image
au format PNG
header ("Content-type: image/png");

// on définit la largeur et la hauteur de notre image
$largeur = 550;
$hauteur = 300;

// on crée une ressource pour notre image qui aura comme largeur $largeur
et $hauteur comme hauteur (on place également un or die si la création se
passait mal afin d'avoir un petit message d'alerte)
$im = @ImageCreate ($largeur, $hauteur) or die ("Erreur lors de la
création de l'image");

// on place tout d'abord la couleur blanche dans notre table des couleurs
(je vous rappelle donc que le blanc sera notre couleur de fond pour cette
image).
$blanc = ImageColorAllocate ($im, 255, 255, 255);

// on place aussi le noir dans notre palette, ainsi qu'un bleu foncé et
un bleu clair
$noir = ImageColorAllocate ($im, 0, 0, 0);
```

```
$bleu_fonce = ImageColorAllocate ($im, 75, 130, 195);
$bleu_clair = ImageColorAllocate ($im, 95, 160, 240);


// on dessine un trait horizontal pour représenter l'axe du temps
ImageLine ($im, 20, $hauteur-40, $largeur-15, $hauteur-40, $noir);

// on affiche le numéro des 12 mois
for ($i=1; $i<=12; $i++) {
    if ($i==1) {
        ImageString ($im, 2, 42, $hauteur-38, $i, $noir);
    }
    else {
        ImageString ($im, 2, ($i)*42, $hauteur-38, $i, $noir);
    }
}

// on dessine un trait vertical pour représenter le nombre de pages vues
ImageLine ($im, 20, 30, 20, $hauteur-40, $noir);

// on affiche les legendes sur les deux axes ainsi que différents textes
(note : pour que le script trouve la police verdana, vous devrez placer
la police verdana dans un repertoire /fonts/)
imagettftext($im, 14, 0, $largeur-70, $hauteur-10, $noir,
"./fonts/verdana.ttf", "Mois");
imagettftext($im, 14, 0, 10, 20, $noir, "./fonts/verdana.ttf", "Nb. de
pages vues");
imagettftext($im, 14, 0, $largeur-250, 20, $noir, "./fonts/verdana.ttf",
"Statistiques pour l'année 2003");

// on parcourt les douze mois de l'année
for ($mois=1; $mois <= 12; $mois++) {
    if ($visite_par_mois[$mois]!="0") {
        // on calcule la hauteur du baton
        $hauteurImageRectangle =
ceil((($visite_par_mois[$mois])*($hauteur-50))/$max_visite));
        if ($mois=="1") {
            // si le mois est janvier, on affiche notre premier baton
            // on affiche le premier baton noir
            ImageFilledRectangle ($im, 42, $hauteur-
$hauteurImageRectangle, 42+14, $hauteur-41, $noir);
            // on affiche le second baton, bleu foncé, qui sera un peu
plus petit que le noir afin de recouvrir une partie du noir
            ImageFilledRectangle ($im, 44, $hauteur-
$hauteurImageRectangle+2, 42+12, $hauteur-41-1, $bleu_fonce);
            // on affiche le dernier baton, bleu clair, qui sera un peu
plus petit que le bleu foncé afin de recouvrir une partie du bleu foncé
(on obtiendra ainsi un effet de dégradé)
            ImageFilledRectangle ($im, 48, $hauteur-
$hauteurImageRectangle+2, 42+8, $hauteur-41-1, $bleu_clair);
        }
        else {
            // si le mois est different de janvier, on affiche les autres
batons
            ImageFilledRectangle ($im, ($mois)*42, $hauteur-
$hauteurImageRectangle, ($mois)*42+14, $hauteur-41, $noir);
            ImageFilledRectangle ($im, ($mois)*42+2, $hauteur-
$hauteurImageRectangle+2, ($mois)*42+12, $hauteur-41-1, $bleu_fonce);
            ImageFilledRectangle ($im, ($mois)*42+6, $hauteur-
$hauteurImageRectangle+2, ($mois)*42+8, $hauteur-41-1, $bleu_clair);
        }
    }
}

// on dessine le tout
Imagepng ($im);
?>
```

Vous noterez quelque chose de très important : en effet, pour faire les dégradés sur les bâtons, nous avons en fait dessiné plusieurs rectangles les uns sur les autres (et donc des rectangles de plus en plus petits).
En effet, on a utilisé GD comme si on utilisait un logiciel de dessin.
En clair, quand on dessine par-dessus quelque chose, et bien ce quelque chose disparaît au profit de la nouvelle forme que l'on vient de dessiner (tout comme des claques sous Photoshop).

Et voici le résultat :



Aller hop, à vos pinceaux maintenant, et surtout n'oubliez pas d'aller jeter un coup d'oeil sur le manuel PHP afin de découvrir toutes les descriptions des instructions utilisées dans ce cours.

Je vous aide un peu, c'est par ici que cela se passe :)

21. Les expressions régulières

Maintenant que nous avons une certaine assurance avec PHP, il serait grand temps de commencer à réfléchir aux expressions régulières.

En effet, les expressions régulières vous permettront d'analyser différentes chaînes de caractères (principalement saisies par un client dans un formulaire) et ainsi vérifier que la chaîne correspond bien à ce que l'on demande.

De plus, les expressions ne servent pas qu'à la vérification de la "conformité" d'une chaîne de caractère, puisqu'elles vont permettront également d'améliorer grandement l'utilisation de votre site.

Les expressions régulières ont été empruntées au système POSIX. De nombreux scripts Perl les utilisent également.
Il s'agit d'un système extrêmement ingénieux permettant de retrouver un mot, voir même une phrase complète (d'une certaine forme) dans un texte, suivant un modèle que l'on aura défini, le pattern.

Par exemple, à l'aide des expressions régulières, vous pourrez facilement vérifier l'adresse email qu'un visiteur dans un champs d'un formulaire de votre site (ceci en testant l'allure de la chaîne de caractère saisie par ce client : on testera entre autre si il y a bien un @ dans cette chaîne de caractères, mais juste un seul, que la chaîne qui suit ce @ est bien un nom de domaine, et que la chaîne qui précède le @ est bien un identifiant valide, c'est-à-dire sans caractères interdits, etc...).

Ou encore, avec les expressions régulières, vous pourrez intégrer dans votre script de news (ou pourquoi pas même votre forum) un système analysant la chaîne de caractère de votre news (de vos messages du forum) en y recherchant des URL, et si elles sont présentes dans votre chaîne, et bien de faire un lien vers cette URL, et ce, tout automatiquement.

Pour la suite du cours, je vais vous parlez seulement des modèles (pattern).

Ce sont ces modèles, qui sont en fait des sortes de masques que l'on applique sur une chaîne de caractères ou un texte pour trouver une suite de caractères bien précise.

Par la suite, il existe diverses fonctions utilisant ces modèles afin d'isoler ces suites de caractères, et d'agir sur eux, et je vous renvoies donc à la documentation pour les étudier (je vous rappelle que ce n'est pas de la fainéantise de ma part de ne pas vous parler de ces fonction, mais il y en a tellement, que le cours serait gigantesque ; et de plus, si vous arriver à créer vos propres modèles, vous n'aurez aucun mal à utiliser ces fonctions avec ces modèles).

Les symboles ^ et $

Dans une expression régulière, le symbole ^ représente le début d'un modèle, et le symbole $ représente la fin d'un modèle.

Voyons de suite quelques exemples :

- ^La globule : identifie une chaîne de caractères qui commence par "La globule"
- enfin$: identifie une chaîne de caractères qui se finit par "enfin"
- ^mot$: identifie une chaîne de caractères qui commence par "mot" et qui fini par "mot", il s'agit donc de la chaîne de caractères "mot" elle-même
- test: identifie une chaîne de caractères qui contient le mot "test"

Jusque là, tout va bien.

En effet, vous pouvez remarquer que si vous n'employer ni ^ ni $ (dernier exemple), cela implique que votre modèle peut se reproduire n'importe où à l'intérieur de votre chaîne de caractères, puisque le modèle n'est ni collé au début de la chaîne (on utilise pas de ^) ni à la fin de la chaîne (on utilise pas de $).

Les symboles *, + et ?

Dans les expressions régulières, les symboles *, + et ? servent à indiquer le nombre de fois qu'un caractère ou une suite de caractères puisse apparaître.
En effet, ces symboles agissent sur les éléments (un caractère ou bien une suite de caractères) qui les précédent.
* indiquera que le caractère (ou la suite de caractères) qui le précède ne pourra apparaître que soit, aucune fois, soit plusieurs fois.
+ indiquera que le caractère (ou la suite de caractères) qui le précède ne pourra apparaître soit, une fois, soit plusieurs fois.
? indiquera que le caractère (ou la suite de caractères) qui le précède ne pourra apparaître soit, aucune fois, soit une et une seule fois.

Voyons alors quelques exemples :

- ab* : identifie une chaîne de caractère contenant un a suivi d'un ou d'aucun b (par exemple, les chaînes "a", "ab", "abbb" respectent ce modèle)
- ab+ : identifie une chaîne de caractère contenant un a suivi d'au moins un b (par exemple, les chaînes "ab", "abb", "abbb" respectent ce modèle)
- ab? : identifie une chaîne de caractère contenant un a suivi d'un ou d'aucun b (par exemple, les chaînes "a", "ab" respectent ce modèle, alors que "abb" ne le respecte pas)
- a?b+$: identifie une chaîne de caractères composée d'aucun ou d'un seul a, suivi d'un ou de plusieurs b, le tout étant situé à la fin de la chaîne

Les accolades {}

Par extensions aux symboles précédents, vous pouvez également des limites qui s'utilisent à l'intérieur d'accolades et qui indiquent le nombre d'occurrence de la chaîne recherchée.
De même, comme pour les symboles *, + et ?; elles affectent l'élément qui les précède.

Voyons tout de suite des exemples :

- ab{2} : identifie une chaîne de caractère composée d'un a suivit d'exactement deux b (en clair, seule la chaîne "abb" passe à travers ce modèle)
- ab{2,} : identifie une chaîne de caractère composée d'un a suivi d'au moins deux b (par exmple, "abb" ou "abbb" respectent ce modèle)
- ab{3,5} : identifie une chaîne de caractère composée d'un a suivi de trois à cinq b ("abbb", "abbbb" et "abbbbb" sont les seules chaînes qui respectent ce modèle)

Sachez que vous devez toujours indiquer le premier chiffre dans vos accolades, alors que le second n'est pas obligatoire.

Vous pouvez également remarquer que l'on peut retrouver la fonction des symboles *, + et ? avec des limites appropriées.
En effet, le symbole :

- * correspond en fait à la limite {0,}
- + correspond en fait à la limite {1,}
- ? correspond en fait à la limite {0,1}

Les parenthèses

Afin de quantifier une chaîne de caractères, vous devez utiliser des parenthèses.

Les exemples étant beaucoup plus parlant que du bla-bla, voici quelques exemples :

- a(bc)* : identifie une chaîne de caractères commençant par un a suivi d'aucune ou de plusieurs séquence de caractères "bc"
- a(bc){1,5}* : identifie une chaîne de caractères commencant par un a suivi d'une à cinq fois la séquence de caractères "bc"

Le symbole |

Le symbole | fonctionne comme l'opérateur booléen OU.

Voyons quelques exemples :

- toto|titi : identifie une chaîne de caractères contenant le mot "toto" ou le mot "titi"
- (b|cd)ef : identifie une chaîne de caractères qui contient la séquence de caractères "bef" ou bien la séquence de caractères "cdef"
- (a|b)*c : indentifie une chaîne de caractères qui contient une alternance de a et de b, chaîne se terminant par un c ("bababbbaac" respecte ce modèle, tout comme "c", "bc" par exemple)

Le symbole .

Le symbole . (le point) représente n'importe quel caractère unique.

Exemple :

- ^.{3,}$: identifie une chaîne de caractères comportant exactement trois caractères

Les crochets []

Les expressions entre crochets ([]) indiquent les caractères qui sont permis à un endroit précis d'un modèle.

Les exemples :

- [ab] : identifie une chaîne de caractères contenant un "a" ou un "b" (ce qui revient au même d'écrire le modèle a|b)
- [a-d] : identifie une chaîne de caractères qui contient les lettres minuscules comprises entre le "a" et le "d" (ce qui équivalent de a|b|c|d ou de [abcd])
- ^[a-zA-Z] : identifie une chaîne de caractères qui commence par une lettre minuscules ou bien par une lettre majuscule
- [0-9]% : identifie une chaîne de caractères qui contient un pourcentage à un seul chiffre
- ,[a-zA-Z0-9]$: identifie une chaîne de caractères qui finit par une virgule suivi d'un caractère (lettre ou chiffre)

Note 1 :

Vous pouvez également lister les caractères que vous ne voulez pas en utilisant le symbole ^ comme premier symbole dans vos crochets.

Exemple : %[^a-zA-Z]% : identifie une chaîne de caractères avec un caractère qui n'est pas une lettre (soit en fait un chiffre ou autres chose, mais pas une lettre) entre deux signes pourcentage

Note 2 :

Pour utiliser les caractères ^, ., [, $, (,), |, *, +, {, dans vos expressions régulières, vous devrez les protéger avec un \ juste avant ceux-ci (car en effet, ces caractères ont une signification spéciale, et donc pour pouvoir les utiliser, il faut les protéger).

Sous - note :

N'oubliez pas que les expressions entre crochets sont une exception à cette règle.
En effet à l'intérieur des crochets, tous les caractères spéciaux, y compris l'antislash , perdent leurs puissances spéciales (en clair, dans les crochets, vous ne devez pas protéger les caractères spéciaux avec un \).

Enfin !!!
A première vue, les expressions régulières ne sont pas simple d'utilisation, mais croyez moi, elles sont extrêmement puissantes.

Et maintenant que vous êtes familier avec les modèles, pourquoi ne pas aller faire un tour du côté de la documentation, afin de voir avec quelles fonctions s'utilisent ces précieuses expressions régulières ?

Ce tutorial est fini, pour votre plus grande joie, ainsi que la mienne :)

22. La programmation objet (concepts fondamentaux)

En programmation, un modèle est une abstraction de la réalité.
Par conséquent, un modèle est une vue subjective de la réalité, mais toutefois, cette vue est toujours pertinente.

En effet, un modèle définit une frontière entre la réalité et la perspective de l'observateur. Il ne s'agit donc pas de la réalité, mais d'une vue très subjective de la réalité.
Par conséquent, un modèle doit permettre de faciliter la compréhension d'un système étudie, mais aussi, de simuler ce système.

Aujourd'hui, en programmation, il existe deux principaux modèle de représentation du monde : le modèle fonctionnel (que vous connaissez déjà) et le modèle objet (que nous allons donc étudier).
Jusqu'à aujourd'hui, vous connaissiez la programmation en PHP avec une approche fonctionnelle.

Afin de mieux comprendre les différences entre ces deux approches, nous allons donc détailler les deux approches, et voir ainsi leurs avantages et leurs inconvénients.

Avec une approche fonctionnelle, vos programmes étaient composés d'une série de fonctions, qui ensemble, assuraient certains services.
Il s'agit d'une approche logique, cohérente et intuitive de la programmation.

Cette approche a un avantage certain que l'on appelle la factorisation des comportements.
En effet, une découpe fonctionnelle intelligente consiste à factoriser certains comportements d'une application, ce qui veut dire que pour créer une fonction d'une application, rien ne vous empêche d'utiliser un autre ensemble de fonctions (qui sont donc déjà écrites).

Mais (il y a toujours un mais ^^), l'approche fonctionnelle a aussi ses défauts, comme par exemple une maintenance complexe en cas d'évolution de votre application.
La factorisation des comportements n'a pas que des avantages. En effet, si on y réfléchit deux minutes, maintenant, nos fonctions sont devenues interdépendantes.

Et ceci implique qu'une simple mise à jour de l'application à un point donné peut impacter en cascade sur d'autres fonctions de notre application.
Naturellement, nous pouvons toujours essayer d'écrire des fonctions les plus génériques possibles mais cela rend le développement de l'application beaucoup plus complexe.

De plus, en cas d'évolution de l'application, même si la structure générale de l'application reste valide, la multiplication des points de maintenance (dus au chaînage des fonctions à cause de la factorisation des comportements) rend l'adaptation extrêmement difficile.
Et dans ce cas, l'application sera alors retouchée dans sa globalité.

Prenons un exemple concret : on a en notre possession une application permettant de gérer une bibliothèque, et suite à la demande du client, notre application doit maintenant être

capable de gérer une médiathèque, c'est-à-dire que l'on pourra emprunter, non seulement des livres, mais aussi, des CD-ROM, des DVD, etc...

Pour faire évoluer notre application, nous devrons faire évoluer les structures de données qui sont manipulées par les fonctions, puis nous devrons adapter les traitements, qui à l'origine, ne manipulaient qu'un seul type de document : les livres.

Nous devrons donc faire évoluer toutes les portions de code qui utilisent la base documentaire, et ce, afin de gérer les données et les actions propres aux différents type de documents.

Exemple : notre application, alors qu'elle ne gérait que des livres, pouvait mettre une sorte de carton jaune à un emprunteur lorsqu'il ne rendait pas son livre dans les temps.

Or, maintenant que notre bibliothèque est devenue une médiathèque, et si l'on désire que le délai avant le fameux carton jaune dépende du document emprunté (le carton jaune arrivera plus ou moins tard, suivant le document emprunté : un livre, un CD-ROM, un DVD, etc...), et bien il va falloir prévoir une règle de calcul pour chaque type de document.

Au final, c'est pratiquement la totalité de l'application qu'il va falloir adapter pour gérer les nouveaux types de documents et les traitements correspondants.

Les modifications que nous avons apportées à notre application de médiathèque, nous permettent de penser qu'une approche objet sera beaucoup plus avantageuse quant à la réutilisation du code déjà écrit.

Mais qu'est ce qu'un objet ?

Un objet est une entité comportant des frontières précises et qui possède une identité (un nom).

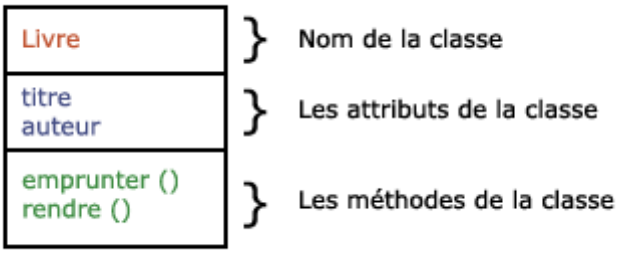
De plus, un ensemble d'attributs caractérisent l'état d'un objet, et l'on dispose d'un ensemble d'opérations (les méthodes) qui permettent d'agir sur le comportement de notre objet.

Un objet est l'instance d'une classe, et une classe, est un type de données abstrait, caractérisé par des propriétés (ses attributs et ses méthodes) communes à des objets et elle permet de créer des objets possédant ces propriétés.

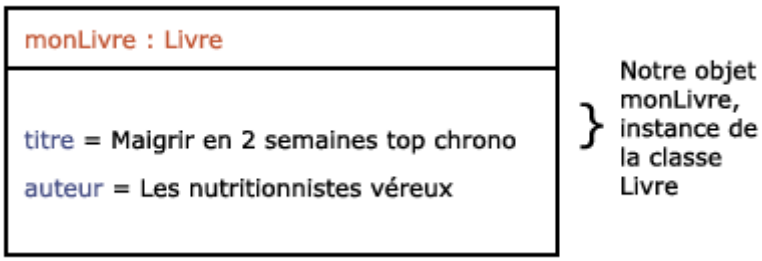
Exemple : prenons le cas de notre bibliothèque.

Créons alors une classe Livre qui va nous permettre de créer des objets "Livre".

On aura alors :



Et une instance de la classe Livre (un objet) :



titre et auteur sont les attributs de notre classe, c'est-à-dire les caractéristiques communes à tous nos objets "Livre".
En clair, tous les livres auront un titre et un auteur.

De plus, notre classe dispose des méthodes emprunter() et rendre() qui s'appliquerons à nos objets "Livre" (on agira alors directement sur notre objet "Livre").

monLivre représente quant à lui, une instance de la classe Livre, il s'agit donc d'un objet qui porte le nom monLivre.

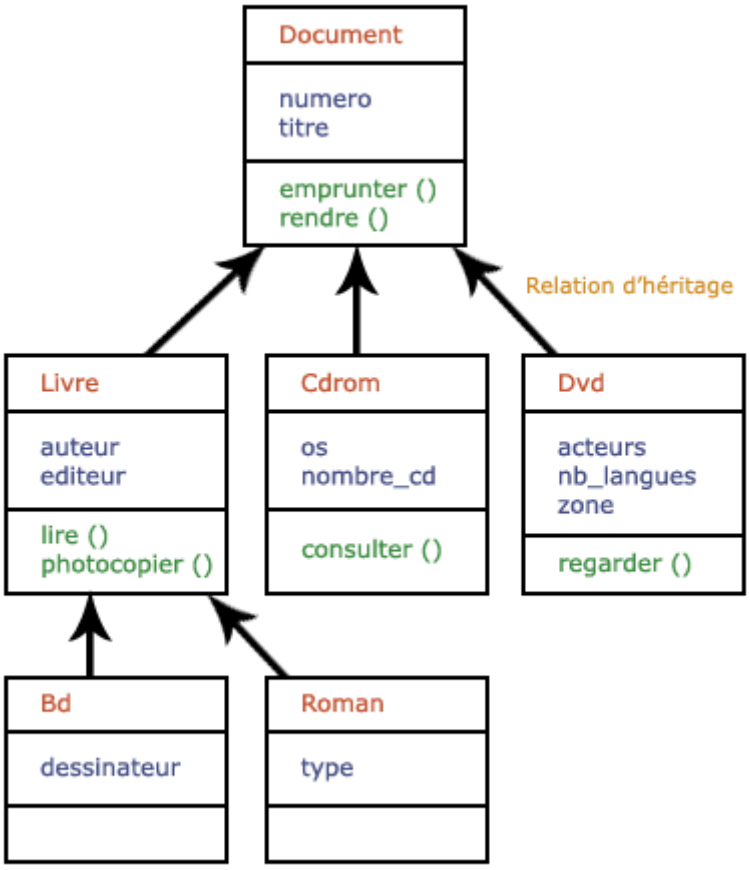
Les autres concepts importants de l'approche objet sont :
- l'encapsulation
- l'héritage (ainsi que le polymorphisme)
- l'agrégation

L'encapsulation consiste à masquer les détails d'implémentation d'un objet, et ce, en définissant une interface. En clair, vous n'avez pas besoin de savoir comment est conçu cet objet à la base pour pouvoir l'utiliser. Une interface est quant à elle, une vue externe d'un objet et elle définit les services accessibles pour modifier le comportement de l'objet.
L'encapsulation facilite l'évolution d'une application car elle stabilise l'évolution des objets. En effet, bous pouvons très bien modifier l'implémentation des attributs d'un objet sans pour autant modifier son interface. Elle garantit de plus l'intégrité des données vu qu'elle permet d'interdire l'accès direct aux attributs des objets (on doit alors passer par des assesseurs).
Un assesseur étant une méthode d'accès pour connaître ou modifier la valeur d'un attribut d'un objet.

L'héritage est un mécanisme de transmission des propriétés d'une classe (ses attributs et ses méthodes) vers une sous-classe (la sous-classe héritant de la classe principale).
Grâce à l'héritage, une classe peut aussi être spécialisée en d'autres classes, afin d'y ajouter des caractéristiques spécifiques (ajout de méthodes par exemple) ou d'en adapter certaines.

Plusieurs classes peuvent aussi être généralisées en une classe qui les factorise, et ce, afin de regrouper les caractéristiques communes d'un ensemble de classes.
La spécialisation et la généralisation permettent de construire des hiérarchies de classes.
L'avantage principal de l'héritage est qu'il vous permet de d'éviter la duplication de code, et il encourage à la réutilisation de même code.

Exemple d'une hiérarchie de classes :

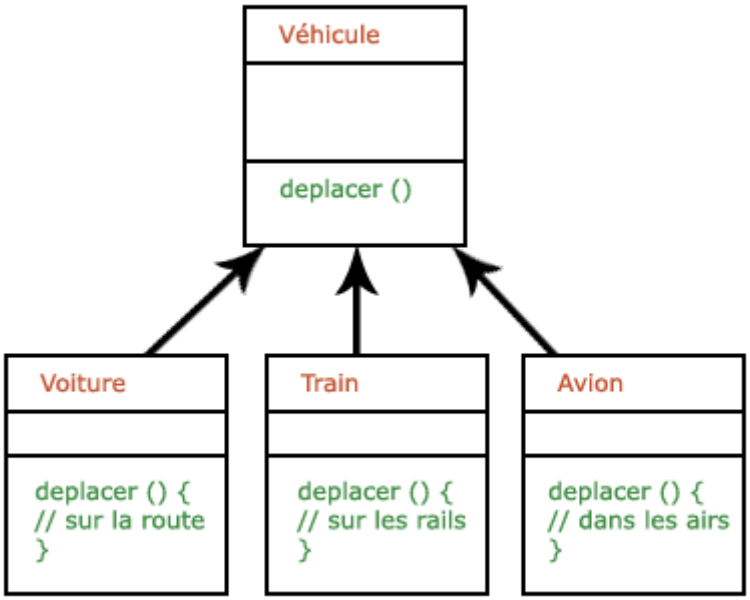


D'après ce graphique, vous pouvez voir que la classe Livre hérite des propriétés de la classe Document.

Ce qui veut dire que les objets Livre auront un auteur, un éditeur, mais aussi un numéro et un titre.
De plus, comme on peut emprunter() ou rendre() un document, on pourra également emprunter() et rendre() un livre (ceci, parce que la classe Livre hérite de la classe Document).
On pourra également lire() ou photocopier() un livre.
En revanche, un document n'aura pas d'auteur ni d'éditeur, et on ne pourra pas le lire, ni même le photocopier (ces propriétés font partie de la classe Livre, or un document n'est pas un livre : c'est un livre qui est un document, et non l'inverse).

Quand au polymorphisme, celui-ci représente la faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes. Il augmente donc la généricité de votre code.

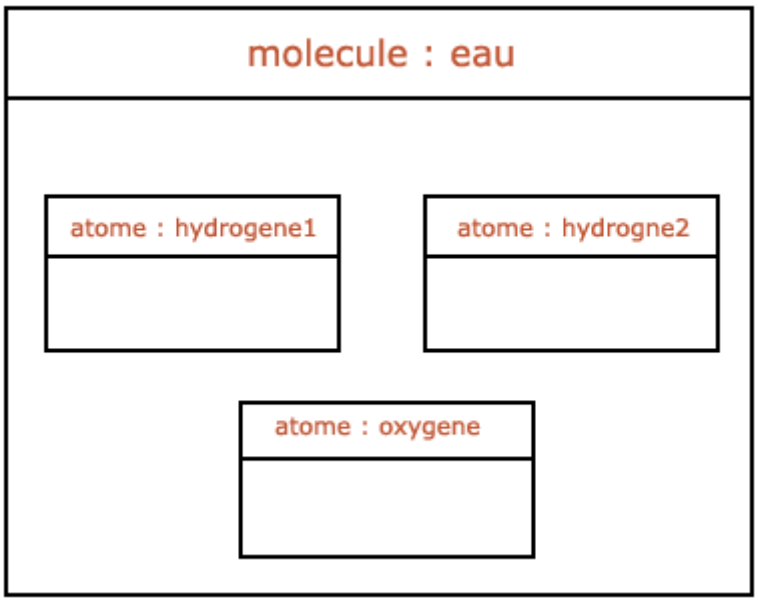
Exemple d'un polymorphisme :



Vous pouvez voir ici que chacune des trois classes (Voiture, Train et Avion) héritent des méthodes de la classe Véhicule.
Ces trois classes, auront donc accès à la méthode deplacer() de la classe Véhicule.
Or, dans ces trois "sous classes", nous avons choisit de redéfinir la méthode deplacer() de la classe Véhicule.
Nous adaptons en fait cette méthode suivant l'objet que nous étudions (un avion se déplace dans les airs, un train sur des rails et une voiture sur la route).

L'agrégation constitue une relation entre deux classes, spécifiant que les objets d'une classe sont des composants de l'autre classe.
Une relation d'agrégation permet donc de définir des objets composés d'autres objets.
L'agrégation permet d'assembler des objets de base, afin de construire des objets plus complexes.

Exemple d'une agrégation :



Vous voyez bien que notre objet eau de type molécule est en fait une combinaison de trois objets :
- hydrogene1 qui est un objet de la classe atome (c'est une instance de la classe atome).
- hydrogene2 qui est un objet de la classe atome (c'est une instance de la classe atome).
- oxygene qui est un objet de la classe atome (c'est une instance de la classe atome).

Notre molécule est donc compose de trois atomes : hydrogene1, hydrogene2 et oxygene.

En conclusion, vous devez savoir que le concept objet est un concept stable et éprouvé.
C'est un concept ancien (Simula, le premier langage de programmation à implémenter le concept de type abstrait à l'aide de classe date de 1967), mais il n'a jamais été autant d'actualité.

A cela, deux raisons principales :
- l'approche fonctionnelle n'est pas adaptée au développement d'applications qui évoluent sans cesse et dont la complexité croit continuellement (plusieurs dizaines de milliers de lignes de code).
- l'approche objet a été inventée pour faciliter l'évolution d'applications complexes.

De nos jours, les outils orientés objets sont fiables et performants (les compilateurs C++ par exemple produisent un code robuste et optimisé).

Mais, malgré les apparences, il est beaucoup plus naturel pour nous, êtres humains, de décomposer un problème informatique sous forme d'une hiérarchie de fonctions atomiques et de données, qu'en terme d'objets et d'interaction entre ces objets.
De plus, le vocabulaire précis est souvent un facteur d'échec important dans la mise en oeuvre d'une approche objet.

C'est pour cela qu'il faut penser objet des le départ, au lieu d'essayer de convertir une série de fonction en une classe.

Apres avoir vu les concepts fondamentaux de la programmation objet, attardons nous sur la programmation objet vu du côté de PHP.

Pour cela, je vous propose de réaliser notre premier classe : une classe qui vous servira à envoyer des mails.

Jusqu'à présent, pour envoyer des mails, vous deviez recopier toujours le même code avec des en-têtes à n'en plus finir.
Tout ceci sera fini avec votre classe d'envois de mails : vous écrirez une fois pour toute votre code, et ensuite, vous réutiliserez ce même code à chaque fois que vous aurez l'envie envoyer des mails.

Afin de commencer sur de bonnes bases, je vous conseille de stocker votre classe de mails dans un fichier PHP, comme par exemple sendmail.class.php, puis d'include ce fichier dans vos scripts lorsque vous en aurez besoin.

Attaquons tout de suite le code de notre classe !

Voici les premières lignes de code de notre classe :

```
<?php
class SendMail {
        var $destinataire;
        var $objet;
        var $texte;
}
?>
```

Comme vous le voyez, la déclaration d'une classe en PHP démarre par le mot clé class suivi du nom de la classe. Nous ouvrons alors une accolade ouvrante qui délimite le début de la déclaration des méthodes et des attributs de la classe (la déclaration de la classe se terminant avec une accolade fermante).

Ensuite, nous joignons 3 variables à notre classe : $destinataire, $objet et $texte.
Ces 3 variables sont en fait les caractéristiques communes des objets (les attributs) que nous allons pouvoir créer grâce à notre classe.
Noter que l'on déclare ces attributs avec le mot réservé var.

En clair, cela veut dire que tous les objets que nous allons créer avec cette classe (des mails en l'occurrence) auront toujours un destinataire, un objet et un texte.

Vous pouvez déclarer autant d'attributs de classe que vous le souhaitez.

Premier exemple d'utilisation :

```php
<?php
// on inclut le code de notre classe
include ("./sendmail.class.php");

// on déclare notre objet (en fait, dans le langage objet, on dit que
l'on crée une instance de la classe SendMail)
$message = new SendMail ();

// on affecte des valeurs aux attributs de notre objet
$message->destinataire = "toto@toto.com";
$message->objet = "Scoop mondial !";
$message->texte = "Voici mon premier mail utilisant une classe mail :)";
?>
```

Noter l'utilisation de l'opérateur -> qui permet d'affecter des valeurs aux attributs de notre objet.
Nous verrons par la suite que cet opérateur nous permet également d'appliquer les méthodes de la classe sur notre objet.

Vous vous doutez bien qu'à ce niveau là, aucun mail n'a été envoyé :)
Cependant, vous pouvez vérifier que notre objet possède bien les attributs que nous lui avons fourni en faisant des :

```php
<?php
echo $message->destinataire; // retournera : toto@toto.com
echo $message->objet; // retournera : Scoop mondial !
echo $message->texte; // retournera : mon premier mail utilisant une
classe mail :)
?>
```

Pour envoyer notre mail, nous allons écrire une méthode pour notre classe qui utilisera la fonction mail de PHP ainsi que les attributs de notre objet.

On aura alors (pour notre classe) :

```php
<?php
class SendMail {
        var $destinataire;
        var $objet;
        var $texte;

        function envoyer() {
                mail ($this->destinataire, $this->objet, $this->texte);
        }
}
?>
```

Détaillons la méthode envoyer().

Notre méthode utilise la fonction de mail de PHP, et elle prend en argument les valeurs des attributs de notre classe.
Cependant, remarquez l'utilisation de $this.
$this indique l'instance courante de notre classe.
Cet objet représente donc notre objet $message que nous utilisons dans notre script.

Vous pouvez écrire autant de méthodes que vous le souhaitez pour votre classe.

Cependant, sachez que vous ne pouvez écrire qu'une seule méthode ayant le même nom que le nom de votre classe (il s'agit en fait du constructeur de la classe qui a un rôle particulier au sein de la classe, nous le détaillerons plus tard).

Envoyons maintenant notre mail grâce à notre méthode envoyer() :

```php
<?php
include ("./sendmail.class.php");

$message = new SendMail ();

$message->destinataire = "toto@toto.com";
$message->objet = "Scoop mondial !";
$message->texte = "Voici mon premier mail utilisant une classe mail :)";

// on applique la méthode envoyer() à notre objet $message grâce à
l'opérateur ->
$message->envoyer();
?>
```

Plutôt pratique non ? :)
En effet, la programmation par objet a cet avantage indéniable, c'est que vous n'étés pas obliger de comprendre comment fonctionne une classe. Du moment que vous savez à quoi servent vos méthodes (leur rôle), vous pouvez les utiliser sans comprendre ce qu'il se passe derrière (le code de la classe en elle-même).

Mais vous pouvez aussi réutilisez vos classes dans d'autres scripts (ce qui reste l'avantage principal de cette forme de programmation).

Voyons maintenant le constructeur d'une classe.

En effet, toute classe contient une méthode dite "constructeur" et cette méthode a toujours le même nom que le nom de la classe en question.

Sans le savoir, lorsque vous avez écrit le code :

```
<?php
message = new SendMail ();
?>
```

Et bien vous avez appelé le constructeur par défaut de la classe SendMail.

Cependant, comme vous l'avez remarqué, notre classe SendMail ne comporte pas de méthode SendMail() (le constructeur d'une classe est la méthode de cette classe qui porte le même nom que le nom de la classe), et bien dans ce cas, PHP utilise un constructeur par défaut qui n'affecte en rien sur les propriétés de notre objet.

Toutefois, vous pouvez vous-même écrire un constructeur dans votre classe.

Mais il faut savoir que le code que vous allez mettre dans votre constructeur va se répercuter sur tous les objets qui seront instances de votre classe.

Prenons un exemple simple :

Imaginer que vous ayez une classe Bonhomme.
Un des attributs de cette classe serait $nom, représentant le nom de notre bonhomme.
Si vous instancier votre classe simplement (en faisant donc un new Bonhomme() tout simple) alors que vous n'avez pas spécifié de constructeur, PHP va vous créer un Bonhomme sans nom (l'attribut $nom de notre objet sera vide).
Cependant, vous pouvez très bien écrire vous-même votre propre constructeur pour donner un nom à votre Bonhomme dès sa création.

Revenons alors à notre classe SendMail, et nous allons lui spécifier un constructeur qui va permettre de définir si le mail à envoyé sera au format texte ou html.

On aura alors :

```php
<?php
class SendMail {
        var $destinataire;
        var $objet;
        var $texte;

        // on ajoute un attribut $entete à notre classe
        var $entete;

        // le constructeur de notre classe qui prend un paramètre
        function SendMail ($type = "texte") {
                if ($type == "texte") {
                        $this->entete = "Content-type: text/plain;
charset=iso-8859-1n";
                }
                elseif ($type == "html") {
                        $this->entete = "Content-type: text/html;
charset=iso-8859-1n";
                }
        }

        function envoyer() {
                mail ($this->destinataire, $this->objet, $this->texte,
$this->entete);
        }
}
?>
```

Vous remarquerez alors que grâce à notre constructeur, nous allons pouvoir rapidement créer des mails au format texte ou bien au format html.

Exemple

```php
<?php
include ("./sendmail.class.php");

// $message sera un mail au format texte
$message = new SendMail ();

// $message2 sera un mail au format texte
$message2 = new SendMail ("texte");

// $message3 sera un mail au format html
$message3 = new SendMail ("html");
?>
```

En résumé, le constructeur de la classe sert souvent à initialiser certains attributs des objets crées à partir de la classe.

Attention, une classe ne peut contenir qu'un seul constructeur.

Ce cours vous a donc présenté quelques rudiments de la programmation par objet en PHP, cependant, vous avez du remarquez que contrairement au premier cours (les concepts fondamentaux), de nombreux concepts n'ont pas étés traités ici.

Ces concepts seront traités dans un futur cours afin de ne pas trop vous embrouillez les idées : trop de notions nouvelles d'un coup peut nuire à la compréhension de cette forme de programmation déjà bien complexe.

23. Les variables dynamiques

Cela ne vous est jamais arrivé de vous demander si il était possible de créer des variables dynamiques ?
C'est à dire des variables ayant un nom changeant au cours du script ?

Et bien c'est tout à fait possible de faire ceci en PHP.

En effet, grâce aux variables dynamiques vous allez pouvoir utiliser des noms de variables qui sont eux mêmes (les noms), variables, ce qui veut dire que vous allez utiliser des noms de variables qui sont affectés et utilisés dynamiquement dans votre script.

D'après ce que l'on vient dire, une variable dynamique serait alors une variable qui aurait comme nom la valeur d'une variable, par exemple $var.
En effet, si nous définissons la variable $hello, avec en valeur la chaîne de caractères Coucou, on aurait la chose suivante :

```
<?php
$var = 'hello';
$hello = 'Coucou';
echo ${$var};
?>
```

Qui affichera :

Coucou

C'est à dire la valeur de la variable $hello.

En utilisant le code :

```
<?php
$var ='hello';
$hello = 'Coucou';
echo $hello;
?>
```

Nous aurions eu exactement le même résultat.

Vous remarquez alors au passage le fonctionnement d'une variable dynamique :
- on initialise une variable ayant une certaine valeur
- on veut utiliser une autre variable ayant comme nom la valeur de notre variable précédente : pour cela, nous déclarons notre variable avec un $, puis avec l'intitulé de notre variable précédent entre deux crochets, ce qui donne : ${$var}

Malgré, cette brève description des variables dynamiques, j ai comme l'impression que vous n'êtes pas du tout convaincu.
Mais à quoi bon peuvent elles bien servir ces variables dynamiques ?

Tout simplement à faire du dynamique :)

Imaginons que l'on ai dans notre script PHP deux variables de type tableau (ou array) et que l'on désire accéder à l'un de ces tableaux.

Au lieu de faire des tests fous pour savoir à quel tableau on veut accéder, il nous suffit de créer une variable de type chaîne de caractères contenant le nom de notre tableau (celui que l'on désire parcourir) et d'utiliser une variable dynamique pour accéder à notre tableau.

Exemple

```
<?php
$tableau1 = array ('test', 'toto', 'titi');
$tableau2 = array ('humpf', 'grmbl');

$var = 'tableau1';
$nb_elements = count (${$var});
for ($i=0; $i<$nb_elements; $i++) {
    // on accede aux éléments du tableau $tableau1
    echo ${$var}[$i].'<br />';
}
?>
```

Ce qui affichera donc :

test
toto
titi

L'exemple peut vous paraître un peu bête, mais regardez le de plus près : en effet, si nous décidons du jour au lendemain de changer le nom de notre tableau (afin de parser un autre tableau), nous n'aurons qu'une seule ligne à modifier :

```
<?php
$var = 'tableau1';
?>
```

(On aurait en fait écrit le nom d'un nouveau nom de tableau)

Alors qu'en n'utilisant pas les variables dynamiques, nous aurions du modifier cette même ligne, mais aussi la ligne concernant le comptage du nombre d'élément de notre tableau mais

aussi la ligne contenue dans notre boucle for (le echo).

Les variables apportent alors leur lot d'avantages :
- la clarté du code
- du code dynamique réutilisable
- des solutions à des problèmes liés à la dynamique d'un script (comme par exemple généré un formulaire contenant un nombre indéfini de champs)

Avant de poursuivre ce cours, sachez également que le nombre de variables dynamiques utilisées est théoriquement sans limites.

Vous pouvez très bien faire, par exemple :

```
<?php
$var  = 'toto';
$toto = 'test';
$test = 'humpf';

echo ${${$var}};
?>
```

Ce qui affichera :

Humpf

Voyons maintenant un cas concret sur lequel il est intéressant d'utiliser les variables dynamiques.
Imaginez que vous ayez un formulaire, et dans la page de traitement de ce formulaire, plusieurs actions sont possibles.

Jusqu'à présent, pour faire votre page de traitement, vous deviez enchaîner une série de if elseif else (ou même mieux en utilisant un case) afin de savoir dans quel cas vous vous trouvé, et de plus, vous deviez écrire un bon paquet de ligne de code pour décrire chaque action.

Tout ceci peut-être largement simplifié grâce aux variables dynamiques.
En effet, imaginons que notre formulaire contienne un champ caché contenant une des valeurs suivantes (au choix) :

- insert
- delete
- update
- select

Voici alors le code d'une page de traitement possible de ce formulaire :

```php
<?php
$champ_cache = $_POST['champ_cache'];

function insert() {
        // code
}

function delete() {
        // code
}
function update() {
        // code
}

function select() {
        // code
}

if ($champ_cache == 'insert' || $champ_cache == 'delete' || $champ_cache
== 'update' || $champ_cache == 'select') {
        $champ_cache();
}
?>
```

Dans ce cas précis, on parlera de fonction dynamique.
Car comme vous pouvez le constater, on utilise la valeur d'une variable pour désigner le nom d'une fonction.

Voyons un autre cas d'utilisation, où là, nous allons pouvoir générer dynamiquement un formulaire.

Générer un formulaire dynamiquement veut dire que la personne utilisant le formulaire pourra choisir, elle-même, le nombre de champs de type text contenu dans ce formulaire, et grâce aux variables dynamiques, nous allons pouvoir récupérer la valeur de ces champs sans connaître auparavant le nombre de champs de ce formulaire (vu que ce nombre est décidé par la personne utilisant le formulaire)

Voici le code :

```php
<?php
// si l'utilisateur soumet le formulaire on affiche la valeur de tous les
champs du formulaire
if (isset($_POST['submit']) && $_POST['submit'] == "Envoyer"){
	// on affiche le nombre de champs du formulaire
	echo 'Nombre de champs : '.$_POST['nb_champs'].'<br />';

	// on affiche la valeur des champs du formulaire
	for ($i=1; $i<=$_POST['nb_champs']; $i++){
		$dynamique = 'champs_'.$i;
		$value = $_POST[$dynamique];
		echo 'Valeur du champ '.$i.' : '.$value.'<br />';
	}
}

// sinon on affiche le formulaire avec la possibilité d'ajout des champs
au formulaire
else{
	// on défini le nombre initial de champs
	if (!isset($_POST['nb_champs'])){
		$_POST['nb_champs'] = 1;
	}
	// si la personne clic sur "un champs en +", on ajoute un champs
	if (isset($_POST['submit']) && $_POST['submit'] == "Un champs en
+"){
		$_POST['nb_champs']++;
	}

	// on affiche le formulaire
	echo '<FORM METHOD="post">';
	// on place un champ caché contenant un entier ayant comme valeur
le nombre de champs du formulaire
	echo '<INPUT TYPE="hidden" NAME="nb_champs"
VALUE="'.$_POST['nb_champs'].'">';

	// on affiche tous les champs du formulaire
	for ($i=1; $i<=$_POST['nb_champs']; $i++){
		echo '<INPUT TYPE="text" name="champs_'.$i.'"><br />';
	}

	// on place un bouton permettant de rajouter un champs
	echo '<INPUT TYPE="submit" NAME="submit" VALUE="Un champs en
+"><br />';
	// on place un bouton permettant de soumettre le formulaire
	echo '<INPUT TYPE="submit" NAME="submit" VALUE="Envoyer">';
	echo '</FORM>';
}
?>
```